Antti Kalliomäki

Koostepalvelujen transaktionhallinta

Sähkötekniikan korkeakoulu

Diplomityö, joka on jätetty opinnäytteenä tarkastettavaksi diplomi-insinöörin tutkintoa varten Espoossa 29.4.2013.

Työn valvoja:

Prof. Jukka Manner

Työn ohjaaja:

DI Janne Eklund

Aalto-yliopisto
Sähkötekniikan
korkeakoulu

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Antti Kalliomäki

Työn nimi: Koostepalvelujen transaktionhallinta

Päivämäärä: 29.4.2013 Kieli: Suomi Sivumäärä:7+47

Tietoliikenne- ja tietoverkkotekniikan laitos

Professuuri: Tietoverkot Koodi: S3022

Valvoja: Prof. Jukka Manner

Ohjaaja: DI Janne Eklund

Liiketoimintaprosessien hallinnan automatisointi vaatii olemassa olevien tietojärjestelmien uudelleenkäyttöä. Tietojärjestelmien toiminnallisuudet voidaan julkaista palveluina, palvelusuuntautuneen arkkitehtuurin mukaisesti. Liiketoiminnallisia tehtäviä suorittavien palvelujen suorittaminen koordinoidusti automatisoi liiketoimintaprosesseja. Koostepalvelu on koordinoitujen palvelukutsujen joukko, joka toteuttaa jonkin liiketoiminnallisesti merkittävän tehtävän. Monet liiketoiminnalliset tehtävät ovat luonteeltaan sellaisia, että ne on suoritettava yhtenä transaktiona. Tällöin koostepalvelun on taattava toiminnallisuutensa transaktionaalisuus.

Tässä työssä on tutkittu koostepalvelun toteuttamista osana prosessinohjausjärjestelmää. Toteutetut koostepalvelut ovat luonteeltaan pitkäkestoisia. Kutsuttavat palvelut sijaitsevat hajautuneessa tietojärjestelmäympäristössä ja ovat toteutuksiltaan heterogeenisiä. Toteutuksen alustana toimii prosessipalvelin, sovelluspalvelin sekä palveluväylä (ESB).

Koostepalvelujen liiketoimintalogiikka on mahdollista toteuttaa koordinoimalla palvelukutsuja BPEL-prosessinkuvauskielellä. BPEL ei kuitenkaan sovellu vaativaan datatransformaatioon palvelukutsujen välillä. Tämän vuoksi BPEL-toteutuksen tueksi on usein tarve toteuttaa paikallisia edustapalveluja ESB-ympäristöön. Koostepalvelun transaktionaalisuus on toteutettavissa myös pitkäkestoisissa koostepalveluissa. Tällöin ratkaisevaa on kompensoivien palvelujen saatavuus sekä kyky kuvata kompensaatio osana koostepalvelun orkestrointia.

Avainsanat: hajautettu transaktio, transaktionhallinta, koostepalvelu, prosessikuvaus, transaktion kompensaatio

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Antti Kalliomäki

Title: Transaction coordination in composite services

Date: 29.4.2013 Language: Finnish Number of pages:7+47

Department of Communications and Networking

Professorship: Data Networks Code: S3022

Supervisor: Prof. Jukka Manner

Instructor: M.Sc. (Tech.) Janne Eklund

Automating business process management often requires that existing information systems can be reused. Functionalities of these legacy systems can be published as services. Service-oriented architecture describes how to do this. Services, that perform business tasks, can be invoked in a coordinated manner to automate business processes. These coordinations are known as composite services. Many of the buusiness processes are transactional in nature. Thus, the composite services must function as transactions as well.

We have investigated how composite services can be implemented as a part of a business process management system. The composite services implement long-running business processes. Services invoked by the composite services are situated in a distributed information system environment and are heterogeneous by implementation. Composite services are implemented in an environment consisting of a process server, application server and an enterprise service bus (ESB).

The business logic of composite services can be implemented by using BPEL language to orchestrate service invokes. It does not, however, suit well to complex data transformations sometimes required for service request messages. Therefore, it is usually necessary to implement local wrapping services to an ESB environment to support BPEL orchestration. The transaction of long-running composite service can be implemented by using compensation. This requires the availibity of compensating services and the ability to define compensation activities as part of service orchestration.

Keywords: distributed transaction, transaction control, composite service, process modelling, transaction compensation

Esipuhe

Haluan kiittää Professori Jukka Mannerta ja ohjaajaani Janne Eklundia työn ohjauksesta, sekä Tuomas Vanhasta sen mahdollistamisesta. Kiitos myös vanhemmilleni opiskelujeni monipuolisesta tukemisesta.

Helsinki, 14.03.2013

Antti Kalliomäki

Sisältö

Lyhenteet

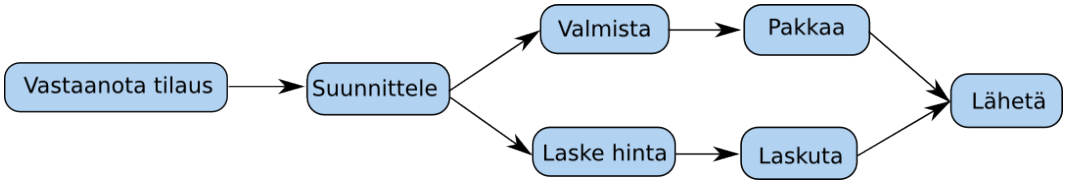
Lyhenteet

ACID	Atomicity, Consistency, Isolation, Durability, transaktion luotettavuusmääreet
2-PC	Two-phase Commit, kaksivaiheinen vahvistuskäytäntö
BPEL	Business Process Execution Language, liiketoimintaprosessin kuvauskielie
BPM	Business Project Management, liiketoimintaprosessien hallinta
ESB	Enterprise Serial Bus, palveluväylä
JSON	JavaScript Object Notation, sanomaformaatti
JTA	Java Transaction API, Java-rajapinta transaktionhallintaan
SCA	Service Component Architecture, sovelluskehitysmalli
SOA	Service-oriented Architecture, palvelusuuntautunut arkkitehtuuri
SOAP	Simple Object Access Protocol, sanomanvälitysprotokolla
TCP/IP	Transmission Control Protocol / Internet Protocol, tiedonsiirtoprotokolla
YAWL	Yet Another Workflow Language, liiketoimintaprosessin kuvauskieli
WSD	Web Service Description, web-sovelluspalvelun rajapinta
WSDL	Web Service Description Language, web-sovelluspalvelun rajapinnan kuvauskieli
XML	Extensible Markup Language, standardoitu merkintäkieli

1 Johdanto

Adam Smith esitti vuonna 1776 julkaistussa teoksessaan *The Wealth of Nations* periaatteen, jonka mukaan useita työvaiheita sisältävä tehtävä on hyödyllistä jakaa yksinkertaisiin osatehtäviin. Erikoistuminen yhteen osatehtävään parantaa työntekijän tehokkuutta, eikä aikaa kulu työtehtävästä toiseen siirtymiseen. [2, luku 1] Smithin esittämä periaate on ollut teollisen tuotannon peruskivenä koko sen historian ajan.

Työn jakaminen yksinkertaisiin osatehtäviin tarkoittaa kuitenkin sitä, että monimutkaisen tuotteen valmistuksessa erillisiä osatehtäviä on suuri määrä. Jotta tuote voidaan valmistaa, nämä osatehtävät on pystyttävä suorittamaan hallitusti ja oikeassa järjestyksessä. Toisin sanoen, valmistamisen prosessia on pystyttävä hallitsemaan. Yleisesti ottaen, prosessi on joukko toisiinsa liittyviä toimintoja ja niiden toteuttamiseen tarvittavia resursseja. Prosessilla voidaan kuvata mitä tahansa toimintaa tai kehityskulkua, jossa syötteet muutetaan tuotoksiksi. Liiketoiminnassa tämän tapahtumaketjun tarkoituksena on usein luoda arvoa asiakkaalle. Organisaatiot ovat kiinnostuneita erityisesti niistä prosesseista, jotka ovat kriittisiä niiden menestymisen kannalta. Näitä kutsutaan usein liiketoiminta-, pää-, tai avainprosesseiksi. [1, s.10-13] Kuvassa 1 on kuvattu päätasolla tilaustuotteen toimitus- ja valmistusprosessiin liittyvät tehtävät tilauksen vastaanottamisesta toimituksen lähettämiseen.

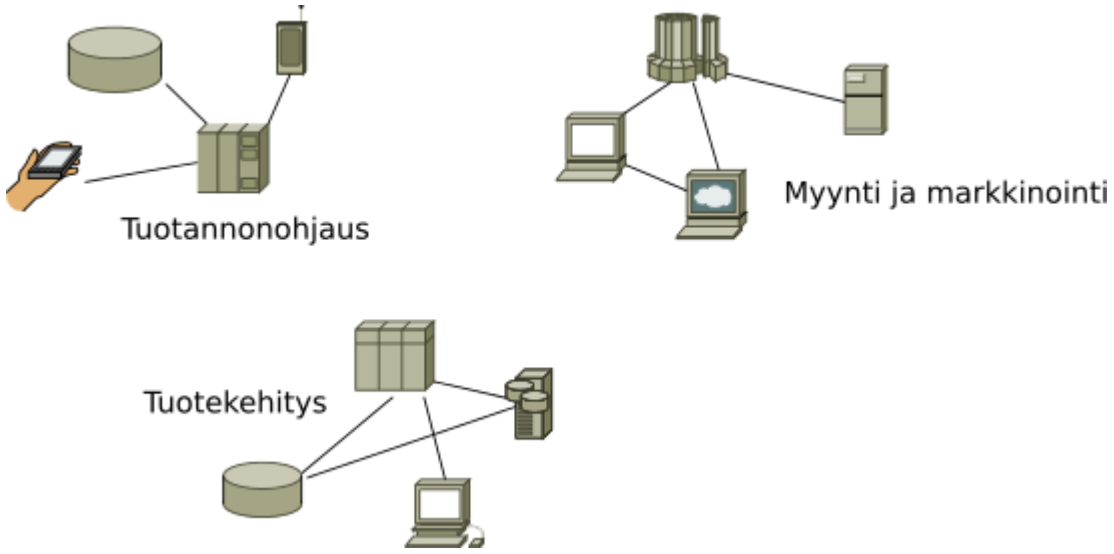


Kuva 1: Tilaustuotteen valmistus- ja toimitusprosessi

Prosessin hallinta vaatii tietoa prosessin tilasta sekä keinoja vaikuttaa prosessin kulkuun tämän tiedon perusteella. Automatisoimalla prosessinhallinta voidaan saavuttaa huomattavia parannuksia tehokkuudessa ja tuottavuudessa. Liiketoimintaprosessin automatisoinnissa yksityiskohtaisella prosessikuvauksella ohjataan työtehtäviä oikeille resursseille. Koska prosessin välivaiheet ovat tiedossa, sen kulkua voidaan seurata ja tarvittaessa niiden kulkuun voidaan puuttua osoittamalla tehtäville lisää resursseja. Prosessin suorituksista voidaan lisäksi kerätä lokitietoa, joka on avuksi prosessin parantamisessa. Liiketoimintaprosessien hallinnan automatisointi tapahtuu tyypillisesti sitä varten kehitetyllä prosessinohjausjärjestelmällä. [20]

Yritykset ja muut organisaatiot käyttävät toimintojensa suorittamisen tukena erilaisia tietojärjestelmiä. Tyypillinen yrityksen tietojärjestelmäympäristö on tyypiltään hajautunut ja heterogeeninen. Hajautunut se on siksi, että yrityksessä on käytössä useita eri tietojärjestelmiä eri osastojen käytössä tai erilaisten tehtävien suorittamista varten. Heterogeenisyys johtuu siitä, että tietojärjestelmät on toteutettu erilaisiin tarpeisiin ja eri aikakausilla, jolloin niiden toteutustekniikat saattavat erota huomattavasti toisistaan. Kuvassa 2 on esitetty liikeyrityksen tietojärjestelmäympäristö, jossa eri osastoilla on käytössään erilliset ja erilaiset tietojärjestelmät.

Näiden tietojärjestelmien tarjoamat toiminnallisuudet sisältyvät organisaation



Kuva 2: Liikeyrityksen hajautunut tietojärjestelmäympäristö

liiketoimintaprosesseihin. Ne saattavat olla organisaation toiminnalle kriittisiä, jonka vuoksi niiden korvaaminen on usein kallista ja aikaavievää. Tälläisestä tietojärjestelmästä käytetään usein nimitystä legacy-järjestelmä. Tämän vuoksi prosessinohjausjärjestelmän on usein pystyttävä uudelleenkäyttämään tietojärjestelmien toteuttamia toiminnallisuuksia.

Monet liiketoimintaprosessit sisältävät tehtäviä, jotka ovat luonteeltaan transaktionaalisia. Transaktio on joukko toimintoja, jotka yhdessä suoritettuna toteuttavat tietyn tehtävän. Toimintojen osittainen suorittaminen ei johda haluttuun lopputulokseen. Transaktiolla on siis vain kaksi haluttua lopputulemaa: se joko suoritetaan kokonaan tai mitään sen osista ei suoriteta. Transaktioille on määritetty neljä vaatimusta, joiden täyttyminen takaa sen, että transaktio on onnistunut:

- Ristiriidattomuus: transaktion osapuolet tekevät vain sallittuja muutoksia resursseihin.
- Jakamattomuus: joko kaikki transaktion osapuolet suorittavat tehtävänsä, tai yksikään niistä ei suorita sitä.
- Pysyvyys: kun transaktio on vahvistettu, sen tulokset ovat pysyviä. Suoritetun transaktion tulosta voi muuttaa vain lisätransaktioilla, joita kutsutaan usein kompensoiviksi transaktioiksi.
- Eristyvyys: suoritetun transaktion tulokset ovat näkyvissä muille transaktioille. Peruutetun transaktion tulokset eivät ole missään vaiheessa näkyvissä muille transaktioille.

Prosessinohjausjärjestelmän kehittämisessä voidaan nähdä kaksi päätehtävää. Ensiksi, prosessi on pystyttävä kuvaamaan tavalla, joka mahdollistaa sen automaattisen suorittamisen. Prosessin suorittaminen on suunniteltava niin, että se ottaa

huomioon tehtävien transaktionaalisen luonteen. Toiseksi, järjestelmän on pystyttävä hyödyntämään hajautetun tietojärjestelmäympäristön resursseja.

Hajautetussa tietojärjestelmäympäristössä viime vuosina voimakkaasti yleistynyt arkkitehtuurimalli on palvelusuuntautunut arkkitehtuuri (service oriented architecture, SOA). Arkkitehtuurin ideana on julkaista tietojärjestelmien toiminnallisuuksia palveluina. Tätä periaatteita noudattavia tekniikoita käyttämällä tietojärjestelmien toiminnallisuudet saadaan käytettäväksi koko organisaation laajuudella. Palvelusuuntautuneen arkkitehtuurin infrastruktuurina nähdään erityisesti palveluväylä (enterprise serial bus, ESB) ja web-sovelluspalvelut (web services).

Kun tietojärjestelmien toiminnallisuudet on julkaistu palveluina, jotka toteuttavat tietyn liiketoiminnallisen tehtävän, liiketoimintaprosesseja voidaan automatisoida kutsumalla näitä palveluja ennalta määrätyssä järjestyksessä. Samoin uusia uudelleenkäytettäviä palveluja voidaan muodostaa toteuttamalla koostepalveluja, jotka muodostavat useasta erillisestä palvelukutsusta. Koostepalveluja toteutettaessa palvelukutsut on toteutettava koordinoidusti, mikä varmistaa niiden oikean suoritusjärjestyksen. Prosessinohjaustuotteissa parhaiten tuettu koordinaatiotapa on palvelujen orkestrointi. Orkestrointi voidaan toteuttaa tehokkaimmin siihen erityisesti kehitetyllä kielellä, kuten Business Process Execution Language (BPEL).

Prosessin tai koostepalvelun transaktionaalisuuden takaaminen vaatii osatehtävien sisällyttämistä yhteen transaktioon. Tämä edellyttää, että osatehtävien transaktiot on pystyttävä hallitsemaan kokonaisuutena, hajautettuna transaktiona. Yleisesti käytetty hajautetun transaktion koordinointitapa on kaksivaiheinen vahvistamiskäytäntö (two-phase commit, 2-PC). Koordinointitavan käyttäminen vaatii kuitenkin tukea kaikilta transaktioon osallistuvilta palveluilta, mikä heterogeenisessa ympäristössä voi olla ongelmallista. Liiketoiminnallisia tehtäviä suorittavat koostepalvelut saattavat myös olla ajallisesti pitkäkestoisia, mikä aiheuttaa teknisiä ongelmia hajautetun transaktion koordinointiin. Tällöin koostepalvelun transaktionaalisuus pystytään osittain toteuttamaan käyttäen hyväksi kompensoivia transaktioita. Kompensoivan transaktion suorittaminen kumoaa loogisesti suoritetun transaktion tulokset.

Tässä työssä tarkastellaan koostepalvelun toteuttamista osana prosessinohjausjärjestelmää. Erityisesti kiinnitetään huomiota koostepalvelun transaktionaalisuuden toteuttamiseen tilanteessa, jossa käytettävissä olevien palvelujen ominaisuudet sekä koostepalvelun tehtävän pitkäkestoisuus estävät transaktiokoordinaattorin käytön. Koostepalvelujen toteuttaminen on pitkälti kiinni käytettävissä olevien palvelujen ominaisuuksista. Tämän vuoksi tässä työssä tarkastellaan myös miten hajautuneen tietojärjestelmäympäristön toiminnallisuudet saadaan koostepalvelutoteutuksen käytettäviksi ja mitkä ovat niiden valmiudet osallistua hajautettuun transaktionhallintaan. Koostepalvelut toteutetaan käyttäen BPEL-kieltä sekä palveluväyläympäristöön toteutettavia välityssovelluksia.

Työn tuloksista selviää, että useimmat hajautuneen tietojärjestelmäympäristön toiminnallisuudet on mahdollista saada koostepalvelutoteutuksen käytettäväksi käyttäen yleisesti käytössä olevia yhteys- ja tiedonsiirtoprotokollia. Tärkeimpänä mahdollistavana teknologiana nähdään web-sovelluspalvelut. Oikein sovellettuina ne mahdollistavat myös hajautettuun transaktionhallintaan osallistumisen. Koostepal-

velutoteutuksien kautta osoitetaan, että palvelun transaktionaalisuus voidaan saavuttaa myös tilanteissa, joissa kutsuttavat palvelut ovat kykenemättömiä osallistumaan transaktiokoordinointiin tai tehtävän pitkäkestoisuus estää sen käyttämisen. Tällöin tärkeäksi tekijäksi nousee palvelukutsujen kompensoitavuus.

Työn rakenne on seuraava. Luvussa 2 esitellään hajautuneen tietojärjestelmäympäristön integraatiotapoja sekä niiden soveltumista prosessinohjausjärjestelmän infrastruktuuriksi. Luvussa kolme käsitellään transaktioita ja hajautetun transaktionhallinnan toteuttamista. Luvussa 4 esitellään koostepalvelutoteutukset sekä tarkastellaan niiden transaktionaalisuuden toteutumista.

2 Koostepalvelut hajautuneessa ympäristössä

Tässä luvussa kerrotaan hajautuneen tietojärjestelmäympäristön integraatiotekniikoista sekä prosessinohjausjärjelmien ja koostepalvelujen käytön edellytyksistä sekä toteutustekniikoista. Ensiksi käsitellään hajautuneen tietojärjestelmäympäristön integrointia käyttäen välikerrosohjelmistoja. Palvelusuuntautunut arkkitehtuuri ja web-sovelluspalvelut esitellään ratkaisuna muuten yhteensopimattomien järjestelmien yhdistämiseksi, myös yli organisaatiorajojen. Tämän jälkeen selvitetään, kuinka integraatio mahdollistavaa prosessinohjauksen tietojärjestelmien kehittämisen niin, että ne nojaavat uudelleenkäytettävien palvelujen hyödyntämiseen. Tällöin tärkeäksi nousee palveluiden koostaminen. Koostamistekniikoista esitellään tarkemmin orkestrointi ja siihen liittyvät teknologiat.

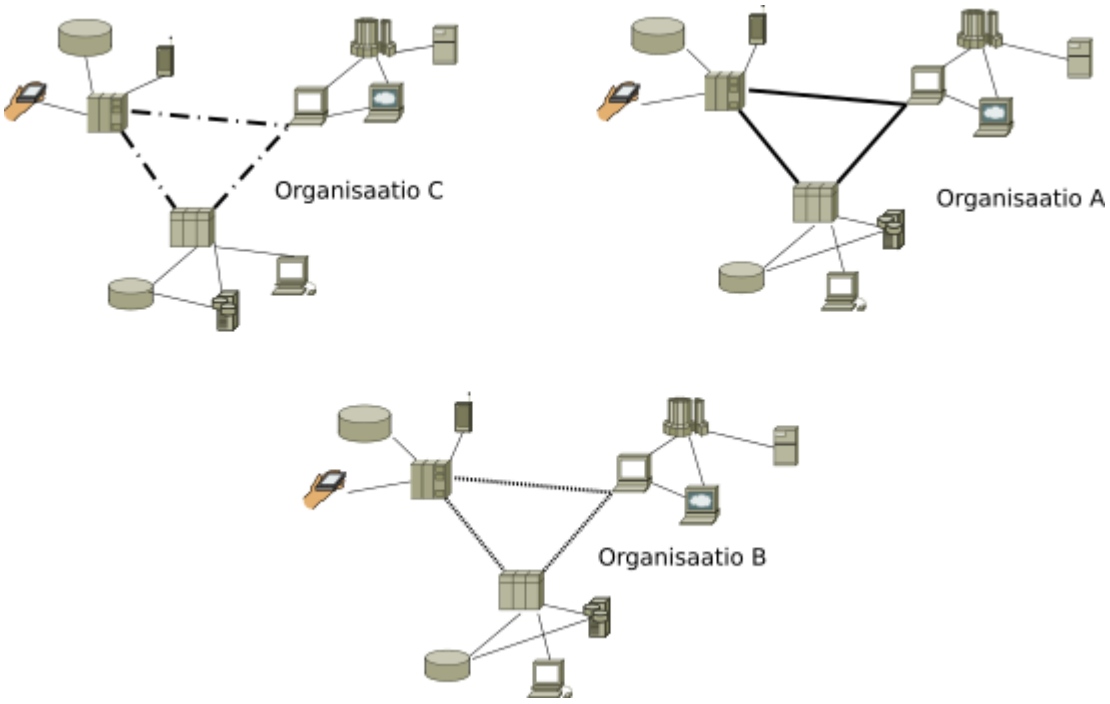
2.1 Hajautuneen ympäristön integraatio

Erillisiä ja toteutustavoiltaan poikkeavia tietojärjestelmiä voidaan yhdistää käyttämällä välikerrosohjelmistoja (engl. middleware). Välikerrosohjelmistojen tärkein tavoite on helpottaa hajautetun tietojärjestelmäympäristön resurssien käyttämistä osana uusia sovelluksia. [30] Tietojärjestelmäympäristön hajautuneisuus ja heterogeenisyys pyritään siis piilottamaan tarjoamalla yhdenmukainen ohjelmointirajapinta kaikkiin ympäristön järjestelmiin. Välikerrosohjelmistot toimivat käyttöjärjestelmien ja tiedonvälitysprotokollien varassa. Usein ne toteuttavat myös yleisiä, sovelluskehitystä tukevia toimintoja, kuten tietoturva- ja transaktionhallintatoimintoja.

Välikerrosohjelmistojen käyttö erillisten tietojärjestelmien yhdistämiseksi luo kuitenkin ongelman organisaatioiden väliselle tasolle. Välikerrosohjelmistoilla voidaan yhdistää erillisiä tietojärjestelmiä, mutta yhden välikerrosohjelmiston ympärille muodostuneet tietojärjestelmäryppäät eivät pysty viestimään keskenään Internetin välityksellä. [17] Kuvassa 3 on kuvattu tilannetta, jossa kolmen organisaation erilaiset välikerrosohjelmistotototeutukset estävät kokonaisintegraation.

Välikerrosohjelmistojen toiminnallisuuksia on usein mahdoton käyttää koostepalvelujen toteuttamiseen erillisten organisaatioiden tarjoamista palveluista. Ensinnäkin, organisaatioiden käyttämät välikerrosohjelmistot ovat usein erilaisia, sillä ne on valittu etusijassa organisaation sisäisten tarpeiden perusteella. Erilaisten välikerrosohjelmistojen integrointi voi olla teknisesti hankalaa ja aiheuttaa korkeita ylläpitokuluja. Toiseksi, välikerrosohjelmistojen sanomia ei useinkaan pystytä välittämään Internetin välityksellä, sillä ne läpäisevät huonosti palomuureja. Lisäksi välikerrosohjelmistoilla toteutetut koostepalvelut ovat luonteeltaan tiukasti sidottuja, joka tekee niiden kehityksestä ja ylläpidosta työlästä ja kallista. [16]

Varsinkin tilanteessa, jossa on tarve automatisoida organisaatiorajojen yli ulottuvia prosesseja, on järjestelmien yhteentoimivuus työlästä ja hankalaa toteuttaa välikerrosohjelmistoja integroimalla. Organisaatioiden käytössä olevat välikerrostoteutukset eivät välttämättä ole yhteensopivia ja organisaatiot eivät välttämättä halua ulkopuolisille pääsyä välikerrosohjelmistoonsa liiketoiminnallisista syistä. Lisäksi välikerrosohjelmistot on perinteisesti suunniteltu lyhytkestoisia vuorovaikutuksia varten, kun taas organisaatioiden väliset vuorovaikutukset ovat tyypillisemmin luon-



Kuva 3: Eri välikerrosohjelmistoilla integroituja tietojärjestelmäympäristöjä.

teeltaan pitkäkestoisia.

2.2 Palvelusuuntautunut arkkitehtuuri

Palvelusuuntautunut arkkitehtuuri (Service-oriented Architecture, SOA) on integraatioarkkitehtuuri, jossa sovelluksien toiminnallisuuksia julkaistaan palveluina. Palveluja voidaan käyttää uusien sovellusten rakennuspalikoina. Arkkitehtuurin mukaiset palvelut ovat usein yksittäistä operaatiota laajempia kokonaisuuksia, jotka toteuttavat tietyn liiketoiminnallisen tehtävän. Ne ovat itsenäisiä sovelluskokonaisuuksia, jotka viestivät löyhästi kytketyllä ja sanomapohjaisella viestintämallilla. [8, luku 2.1]

Löyhällä kytkennällä tarkoitetaan palvelun käyttäjän ja tarjoajan tietämisen rajoittamista toisesta osapuolesta. Tämä tapahtuu muodostamalla palvelulle rajapinnan, joka määrittelee palvelun toiminnan ja piilottaa sen sisäisen toiminnan. Rajapinta määrittelee palvelusta vain ne toiminnot, jotka ovat merkityksellisiä kanssakäymiselle käyttäjien kanssa, mutta piilottaa toteutuksen yksityiskohdat. Tosielämässä löyhän kytkennän saavuttaminen voi olla vaikeaa, sillä usein palvelun tarjoajan ja tilaajan välisen yhteyden toteuttaminen tietoturvallisesti vaatii toteutustapatason ymmärrystä ja yhteistyötä. Samoin osallistuminen hajautettuun transaktionhallintaan voi edellyttää tietoa käytetystä transaktionhallinnasta sekä poikkeustilanteiden seuraksista. Palvelun käyttäjän tarjoajan on jaettava ymmärrys myös kanssakäymisessä käytettävästä tietomallista.

Palvelusuuntautuneen arkkitehtuurin päämotivaationa on, että uudelleenkäyttämällä ja yhdistelemällä olemassaolevia tietojärjestelmien toiminnallisuuksia pys-

tytään nopeasti ja edullisesti kehittämään uusia toiminnallisuuksia, jotka puolestaan mahdollistavat yrityksen sopeutumaan nopeasti muuttuviin liiketoiminnallisiin muutoksiin. Palvelusuuntautuneen arkkitehtuurin käyttö tuo seuraavia etuja:

- Olemassa olevien tietojärjestelmien toiminnallisuus voidaan julkaista palveluna, joka mahdollistaa lisähyödyn saamisen jo tehdyistä investoinneista.
- Hajautetun järjestelmän integraatio voidaan toteuttaa palvelumääritelmien perusteella, välittämättä siitä miten niiden tarjoama toiminnallisuus on toteutettu. Tämä eristää erillisten järjestelmien toteutustason integraatiotasosta, jolloin monimutkaisesta järjestelmäkokonaisuudesta tulee hallittavampi.
- Sovellusten kehittäminen uudelleenkäyttämällä ja koostamalla olemassa olevia palveluja nopeuttaa kehitystyötä huomattavasti. Samalla uuden järjestelmän kehittämiskustannukset alenevat.

2.3 Web-sovelluspalvelut

Web-sovelluspalvelut (engl. web services) on kokoelma teknisiä määritelmiä järjestelmien välisen viestinnän standardoimiseen. Web-sovelluspalvelut noudattaa web-aikakauden perusperiaatteita. Se pohjautuu palvelin/asiakas-arkkitehtuuriin ja käyttää hyväkseen hyvin tuettuja, avoimia standardeja kuten XML, URL ja HTTP. Web-sovelluspalvelut soveltuvat palvelusuuntautuneen arkkitehtuurin toteuttamiseen, sillä ne ovat itseselittäviä, modulaarisia sovelluksia, jotka ovat kutsuttavissa Internetin yli. Web-sovelluspalvelu voidaan toteuttaa mille ohjelmistoalustalle tahansa, mikä tukee palvelusuuntautuneen arkkitehtuurin vaatimusta palveluiden saatavuudesta. [8, s.37]

Web-sovelluspalvelun rajapinta kuvataan sen palvelumääritelmässä (engl. Web service description, WSD). WSD on koneluettava määritelmä palvelun rajapinnasta, ja se on kirjoitettu XML-pohjaisella Web service description language (WSDL) -kielellä. WSDL-kuvauksen parametreja ovat *PortType*, *Binding* ja *Port*. *PortType* määrittelee web-sovelluspalvelun operaatioiden toiminnallisuudet ja niiden käyttämät tietomallit. *Binding* sisältää tietoa, jota vaaditaan palvelun kutsumiseen, kuten käytettävän viestiprotokollan. *Port* määrittelee palvelun käyttämän verkko-osoitteen. WSDL:n suurin hyöty on se, että sillä pystytään kuvaamaan palveluita alustariippumattomasti. Näin yhdellä alustalla toimivan palvelun julkaiseman WSDL-kuvauksen avulla voidaan generoida palvelua kutsuva sovellus toiselle ohjelmistoalustalle. WSDL tukee kuvauksen laajentamista sisältämään esimerkiksi palvelun suojauksen ja transaktionaalisuuden.

Web-sovelluspalvelun vuorovaikutus tapahtuu usein käyttämällä SOAP-sanomia. [18] Simple Object Access Protocol (SOAP) [33] on XML-pohjainen sanomanvälitysprotokolla. SOAP-sanoman otsakkeilla voidaan määritellä, miten viestiä käsitellään välikerrosohjelmistossa. Tekstipohjaisuutensa vuoksi SOAP-viestejä pystytään välittämään useimmissa ympäristöissä ja tiedonsiirtoprotokollilla. Käytetyin tiedonsiirtoprotokolla on HTTP, mutta myös useisiin välikerrosohjelmistoihin on kehitetty SOAP-tuki. Vaikka SOAP ei perusmuodossaan tue tiedonvälityksen suojausta

tai transaktionalisuutta, nämä pystytään toteuttamaan käyttämällä hyväksi SOAP-otsakkeiden lisäkenttiä.

Korkean abstraktiotason vuorovaikutusprotokollaksi web-sovelluspalvelujen erityisominaisuus on se, että sillä on hyvin laaja-alainen tuki IT-teollisuudessa. Tämä on itsessään merkittävä tekijä sen menestyksessä. [7] Koska web-sovelluspalvelun määritelmältä on luonteeltaan kokoelma standardeja, palvelut voivat poiketa merkittävästi toisistaan riippuen siitä, mitä standardeja, tai niiden versioita, on valittu käytettäväksi. Paremman yhteensopivuuden saavuttamiseksi Web Services Interoperability Organization (WS-I) on määritellyt web-sovelluspalveluprofiileja, jotka kuvaavat tiettyjä standardikombinaatioita. [22] Ohjelmistotuotteisiin voidaan tällöin toteuttaa tietyn profiilin toteuttavia palveluja, joiden tarkat ominaisuudet ovat tällöin tarkasti käyttäjien ja asiakkaiden tiedossa.

2.4 Prosessinohjauksen tietojärjestelmät

Prosessinohjausjärjestelmä on tietojärjestelmä, jota voidaan käyttää liiketoimintaprosessin automatisointiin. Ohjausjärjestelmää varten prosessi on kuvattava riittävällä tarkkuudella kaikista automatisointiin liittyvistä näkökulmista. Tärkeimpiä kuvattavia asioita ovat prosessiin sisältyvät yksittäiset tehtävät, niiden suoritusjärjestys sekä tehtävien välillä kulkeva tieto.

Prosessinohjausjärjestelmä toteutetaan usein käyttämällä siihen tarkoitettua sovellusintegraatioympäristöä. Ympäristö sisältää kehitystyökalut, sanomanvälitysjärjestelmän olemassa olevien tietojärjestelmien kanssa viestimiseen sekä esimerkiksi tietomallien muuntamiseen tarvittavan toiminnallisuuden. [31, s.57] Prosessinohjausjärjestelmä muodostaa keskuksen, jota kautta kaikki prosessiin osallistuvat tietojärjestelmät viestivät. Se myös sisältää mahdollisuuden tallentaa pysyvään muistiin sovelluksilta saatua tietoa, ja välittää sitä eteenpäin muille sovelluksille. Nämä seikat vähentävät tarvetta toteuttaa suoria yhteyksiä sovellusten välille, ja laskee huomattavasti ylläpidettävien integraatioiden määrää.

Tyypillisesti prosessinohjausjärjestelmät perustuvat palvelusuuntautuneen arkkitehtuurin käyttöön. Palvelusuuntautuneessa arkkitehtuurissa prosessinohjausjärjestelmän käytössä on palveluita, jotka toteuttavat tietyn liiketoiminnallisen tehtävän. Prosessinohjausjärjestelmällä toteutettu liiketoimintaprosessin automatisointi on usein yhdistelmä palvelukutsuja olemassa oleviin tietojärjestelmiin ja inhimillistä panosta vaativia työtehtäviä.

2.5 Palveluiden koostaminen

Palvelusuuntautuneen arkkitehtuurin päätavoite on tarjota erillisiä palveluja, joista voidaan muodostaa koostepalveluja. [12, luku 3] Koostepalveluiden avulla puolestaan voidaan automatisoida monimutkaisia liiketoimintaprosesseja. Palveluiden koostamisen myötä saadaan suurin hyöty palvelusuuntautuneesta arkkitehtuurista.

Palvelukoosteiden toteutettavuus riippuu käytössä olevien palvelujen ominaisuuksista [19]. Ensinnäkin, jotta koostepalvelulla voidaan toteuttaa jokin liiketoiminnallinen tehtävä, myös yksittäisten palvelujen on toteutettava liiketoiminnalli-

sesti itsenäinen toiminnallisuus, jota on mahdollista sekä mielekästä käyttää osana koostepalvelua. Toiseksi, palvelu on oltava toteutettu tekniikalla, joka mahdollistaa sen toiminnallisuuden hyödyntämisen osana palvelusuuntautunutta arkkitehtuuria.

Koostamiseen on kaksi erilaista lähestymistapaa: *orkestrointi* ja *koreografia* [19]. Orkesteroinnin keskiössä on prosessikuvauksen tai koostelogiikan sisältä koordinaattori, joka hallinnoi liiketoimintapalveluja ja koordinoi niiden tarjoamien toimintojen kutsumista. Kutsutut palvelut eivät tiedä olevansa osa orkestroitua koostepalvelua, vaan kutsuttavien palvelujen ja toimintojen järjestys on määritelty eksplisiittisesti koordinoivassa prosessissa. Keskitettyä orkestrointia voidaan kutsua suoritettavaksi liiketoimintaprosessiksi.

Koreografia ei sisällä keskitettyä koordinaattoria, vaan siinä koostamisen logiikka on hajautettu liiketoimintapalveluihin. Jokainen palvelu tietää, milloin sen on suoritettava toimintonsa ja minkä muiden palveluiden kanssa olla yhteydessä.

Eron tekeminen liiketoimintapalvelu ja -prosessitoteutuksen välillä ei ole yksiselitteistä, ei niiden tarjoamien palveluiden suhteen eikä myöskään teknisen toteutuksen puolesta. Liiketoimintaprosessi voi muodostua usean liiketoimintapalvelun koosteesta. Prosessitoteutus voidaan kuitenkin julkaista rajapinnalla samaan tapaan kuin liiketoimintapalvelu. Tällöin prosessitoteutusta voidaan käyttää palveluna, ja osana vielä monimutkaisempaa liiketoimintaprosessia.

Liiketoimintapalvelut voidaan jakaa kahteen kategoriaan kompleksisuutensa suhteen [19]:

- Diskreettin palvelun toiminnot voidaan suorittaa yksittäisellä vuorovaikutuksella eikä toiminnallisuutta voida jakaa pienempiin osiin. Diskreetti palvelu voi olla synkroninen tai epäsynkroninen.

- Koostepalvelun toiminnallisuuden suorittamiseen tarvitaan useampi kuin yksi vuorovaikutus.

Jotta koostepalvelulla voidaan automatisoida liiketoimintaprosessi, sen on joissain tapauksissa jouduttava valitsemaan ajonaikaisesti usean erilaisen toimintaskenaarion väliltä. Niistä jokainen saattaa sisältää erilaisen joukon kutsuttavia palveluita. Nämä skenaariot ja niihin sisältyvät palvelukutsut ovat koostesuunnittelun olennaisia osia. Niihin on kiinnitettävä huomiota koostepalvelun suunnittelussa, jotta koostepalvelu saavuttaa halutun lopputuloksen kaikista mahdollista lähtökohdista. [12, luku 3]

Palvelusuuntautuneen arkkitehtuurin täysimittainen hyödyntäminen edellyttää, että kaikki sen piirissä olevat palvelut on toteutettu ottaen huomioon arkkitehtuurin vaatimukset. Organisaation it-infrastruktuurin on tuettava palvelusuuntautunutta arkkitehtuuria, sillä palvelukoosteen arkkitehtuuri on hyvin riippuvainen palveluiden suoritusympäristön tietoturva-, transaktionhallinta- ja sanomavälitysominaisuuksista. Palveluiden on oltava käytettävissä koko organisaation laajuudella. Se edellyttää usein standardoitujen yhteysprotokollien käyttöä, jolla varmistetaan palvelujen uudelleenkäytettävyys ja yhteensopivuus. [12, luku 3] Palveluja suunniteltaessa ja toteutettaessa on ajateltava palvelun uudelleenkäyttöä yksittäisen käyttötapauksen sijaan. Muuten on vaarana ajautua tilanteeseen, jossa jokaiseen käyttötapaukseen

toteutetaan omat palvelunsa, jotka ovat lähes identtisiä toisten käyttötapausten palveluiden kanssa.

2.6 Koostepalvelun orkestrointi

Monimutkaisia koostepalveluja voidaan kehittää yleiskäyttöisillä ohjelmointikielillä, kuten Java ja C++. Ne eivät kuitenkaan tarjoa valmiita korkean tason abstraktioita, joilla voi helposti kuvata palvelun koostumista useista peräkkäisistä, rinnakkaisista tai ehdollisista palvelukutsuista. Koostepalveluiden orkestrointiin kehitettyjä kieliä ovat muun muassa Business Process Execution Language (BPEL) sekä Yet Another Workflow Language (YAWL) [34]. Koreografiointiin kehitetyistä kielistä Web Service Choreography Description Language (WS-CDL) eteni standardoimisehdokkaaksi asti, mutta sitä kehittänyt W3C:n työryhmä lopetti toimintansa vuonna 2005.

2.6.1 Business Process Execution Language

Web Services Business Process Execution Language (WS-BPEL) [5] [4] on OASIS-ryhmän standardoima XML-pohjainen kieli. Sitä voidaan käyttää orkestroimaan monimutkaisia prosessitoteutuksia sekä palvelukoosteita, joissa kutsutaan useita eri liiketoimintapalveluita ennalta määritellyssä järjestyksessä [19]. Tälläisissä koostepalveluissa voidaan käyttää sekä synkronisia että epäsynkronisia palvelukutsuja. Koostepalvelut voivat olla luonteeltaan pitkäkestoisia ja kutsujen välillä on pystyttävä säilyttämään tilatietoa. Tilatiedon avulla huolehditaan, että prosessi tai palvelu suoritetaan oikein, sekä välitetään tietoa palvelukutsusta toiseen. Lisäksi BPEL-kielellä pystytään kuvaamaan koostepalvelun toimintaa vika-, ja muissa poikkeustilanteissa.

Kirjoitushetkellä uusin versio on huhtikuussa 2007 julkaistu WS-BPEL 2.0. Standardista käytetään yleisesti lyhennettä BPEL, jos ei ole tarvetta viitata tiettyyn versioon. BPEL-kielen standardi ei sisällä graafista notaatiota.

BPEL mahdollistaa palvelukoosteen kuvaamisen niin, että tuloksena on joko *suoritettava prosessi* tai *abstrakti prosessi*. Suoritettava prosessi määrittelee, missä järjestyksessä ja millä logiikalla sovelluspalveluja kutsutaan, sekä niiden kutsu- ja vastaussanomat. Prosessin ulkopuolisiin toimijoihin, kuten kutsuttaviin palveluihin, viitataan prosessikumppaneina. Suoritettava prosessi voidaan suorittaa siihen soveltuvassa ympäristössä, jota usein kutsutaan prosessimoottoriksi. Suoritettava prosessi muodostaa uuden web-sovelluspalvelun, joka on kooste olemassaolevista palveluista. Tämä on BPEL-kielen pääasiallinen käyttötarkoitus.

Abstrakti prosessikuvaus sisältää ainoastaan sovellusten välillä liikkuvat viestit. Se ei kerro prosessivuon sisäisiä yksityiskohtia, eikä se siksi ole suoritettavissa. Abstrakti prosessikuvaus on koreografia-paradigman mukainen. Abstraktia prosessikuvausta käytetään usein kahdessa eri tyyppisessä tilanteessa: kun halutaan kuvata sovelluspalvelun käyttäytymistä ilman tietoa siitä, mihin liiketoimintaprosessiin se ottaa osaa, tai, kun määritellään vuorovaikutusprotokollaa ja halutaan kuvata tarkasti jokaisen osapuolen ulkoinen käytös. [3]

Aktiviteetit ovat BPEL-kielen päärakennuspalasia. Aktiviteetteja on kahta päätyyppiä. Jäsentelyaktiviteetit voivat sisältää toisia aktiviteetteja ja määrittävät esi-

merkiksi niiden suoritusjärjestyksen. Perusaktiviteetit puolestaan suorittavat vain yhtä tiettyä tehtävää.

BPEL-kielessä on kolme tärkeää perusaktiviteettia, joiden tarkoitus on mahdollistaa viestinvälitys kutsuvan ja kutsuttavien sovellusten kanssa.

1. *Vastaanottoaktiviteetti* (receive activity) vastaanottaa viestejä orkestraation ulkopuoliselta taholta. Se on linkitetty tiettyyn prosessikumppanin tarjoamaan ulkoiseen operaatioon. Vastaanottoaktiviteetti voi alustaa uuden orkestraation, eli toimia sen käynnistävänä aktiviteettina, tai toimia osana jo olemassaolevaa orkestrointia.

2. *Vastausaktiviteettia* (reply activity) käytetään usein vastaanottoaktiviteetin parina. Sitä käytetään lähettämään viesti prosessikumppanille. Useissa liiketoimintatapauksissa kumppani on sama, jolta prosessin käynnistänyt vastaanottoaktiviteetti on vastaanottanut pyyntösanoman. Lähetettävä viesti voi sisältää tiedon tehtävän onnistumisesta sekä sen suorituksen tuloksia.

3. *Kutsuaktiviteetti* (invoke activity) kutsuu web-sovelluspalveluja. Kutusaktiviteetti sisältää attribuutteja, jotka määräävät mitä prosessikumppanin operaatiota kutsutaan. Kutsuaktiviteetti voi olla joko yksisuuntainen eli asynkroninen, jolloin prosessin suorittamista jatketaan odottamatta vastausta palvelulta, tai kaksisuuntainen eli synkroninen, jolloin prosessin suoritus pysähtyy, kunnes kutsuaktiviteetti saa vastauksen palvelulta.

Perusaktiviteettien lisäksi BPEL sisältää lukuisia jäsentelyaktiviteetteja, jotka mahdollistavat perusaktiviteettien järjestelyn niin, että ne toteuttavat halutun liiketoimintalogiikan.

1. *Jonoaktiviteetin* (sequence activity) sisältämät perusaktiviteetit suoritetaan peräkkäin niin, että seuraavan aktiviteetin suoritus ei ala ennen kuin edellisen aktiviteetin suoritus on loppunut.

2. *Jos-muuten aktiviteetti* (if-else activity) toimii kuten ehtolausekkeet useimmissa ohjelmointikielissä. Se mahdollistaa täsmälleen yhden vaihtoehtoisen prosessihaaran suorittamisen monista vaihtoehdoista. *Jos*-haarojen ehtolausekkeet arvioidaan yksi kerrallaan, ja ensimmäinen ehdon täyttävä haara valitaan suoritettavaksi. Jos mikään ehtolausekkeista ei toteudu, suoritetaan prosessin *muuten*-haara. Ehdolausekkeissa voidaan käyttää XPath-kieltä, jota käytetään osoittamaan XML-dokumenttien tiettyihin osiin [6].

3. *Valinta* (choice) sisältää vaihtoehtoisia suorituspolkuja. Suorituspoluista valikoituu suoritettavaksi se, jonka ehtolause on tosi. Valinta-aktiviteetti voi sisältää mielivaltaisen määrän erillisiä suorituspolkuja. Suorituspolun aloittaa *tapaus-elementti* (case element) ja sitä seuraavat suorituspolkuun kuuluvat aktiviteetit.

4. *Toistoaktiviteetit* (repetitive activies), *while*, *repeatUntil* ja *forEach*, mahdollistavat aktiviteetin tai aktiviteettien suorituksen toistamisen. While ja repeatUntil toistavat suorittamista, kunnes niiden ehtolauseke on tosi. Erona on

se, että repeatUntil-aktiviteetissa sen sisältämät aktiviteetit suoritetaan vähintään kerran. ForEach puolestaan toistaa suorituksen ennalta määritetyn lukumäärän.

5. *Rinnakkaisaktiviteetissa* (flow activity) sen sisältämät aktiviteetit käynnistetään yhtä aikaa ja niiden suoritus on rinnakkaista. Jos aktiviteetit ovat toisistaan riippuvaisia, niiden suorittamista voidaan synkronoida yhdistämällä ne toisiinsa linkeillä.

6. *Asetusaktiviteetti* (assign activity) sisältää kopiointioperaatioita, joilla voidaan kopioida muuttujien sisältöä toisiin muuttujiin. Tämän avulla esimerkiksi vastaanottoaktiviteetin vastaanottamasta viestistä voidaan erotella kutsuaktiviteettien vaatimia tietoja.

7. *Poikkeuskäsittelijä* (fault handler) voidaan kytkeä joko koko prosessiin tai sen osaan. Poikkeuskäsittelijä sisältää yhden tai useampia *catch*-elementtejä ja korkeintaan yhden *catchAll*-elementin. Catch-elementit liittyvät yksittäisiin aktiviteetteihin, ja ne käynnistävät määritellyt aktiviteetit, jos aktiviteetilta saapuu poikkeusilmoitus. CatchAll-elementti sisältää oletusaktiviteetit, jotka käynnistetään minkä tahansa poikkeuksen yhteydessä.

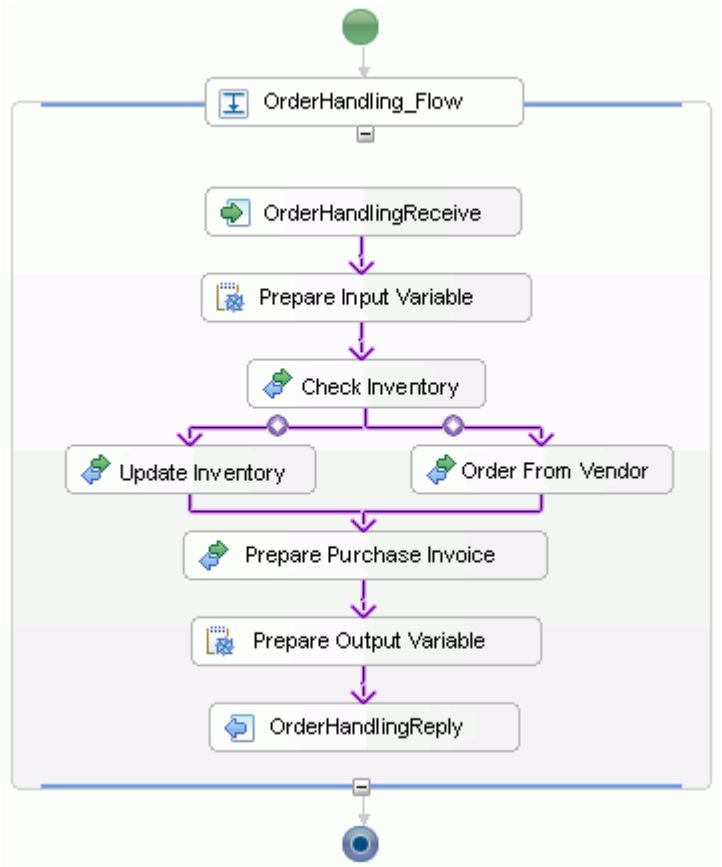
8. *Kompensaatiokäsittelijä* (compensation handler) sisältää ne aktiviteetit, jotka toteuttavat sen prosessin osan kompensaation, johon kompensaatiokäsittelijä on kytketty. Kompensaatiokäsittelijä voidaan kytkeä yksittäiseen aktiviteettiin, koko prosessiin tai sen tiettyyn osaan käyttäen laajuusrajausta.

9. *Laajuusrajausta* (scope) voidaan käyttää jakamaan liiketoimintaprosessi osiin. Jokainen laajuusrajaus voi määritellä muun muassa omat prosessikumppaninsa, prosessimuuttujansa sekä kompensaatio- ja poikkeuskäsittelijänsä.

BPEL-standardi ei sisällä graafisen notaation määrittelyä, mutta useat kehitystyökalut sisältävät mahdollisuuden orkestroida koostepalveluja tai liiketoimintaprosesseja BPEL-kielellä graafisesti. Työkaluja ovat muun muassa IBM WebSphere Integration Designer ja Oracle BPEL Process Manager [35]. Kehitystyökalut luovat graafisen esityksen pohjalta standardin mukaista BPEL-koodia. Osa kehitystyökaluista sisältää tuotekohtaisia laajennuksia prosessikuvaukseen, jonka osalta ohjelman tuottama koodi ei ole standardin mukaista. Kuvassa 4 on WebSphere Integration Designer -kehitystyökalulla graafisesti luotu prosessikuvaus, jonka työkalu tallentaa standardin mukaisena BPEL-kielenä.

BPEL ei ole yleiskäyttöinen ohjelmointikieli, ja sen käytössä on tämän vuoksi monia rajoitteita. Ongelmallisiksi havaittuja rajoitteita on esimerkiksi poikkeustenhallinnassa ja datatransformaatiossa [16]:

- Poikkeuskäsittelijään kytketty catchAll-elementti ei erottele poikkeustyyppejä. Tilanteessa, jossa yksi kutsutuista web-sovelluspalveluista on tavoittamattomissa, kutsuun liitetty catch-elementti ei laukea, sillä se havaitsee ainoastaan



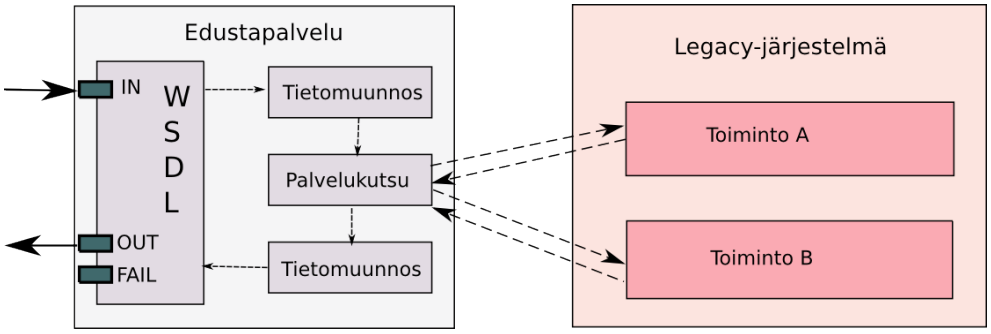
Kuva 4: WebSphere Integration Designer -kehitystyökalun graafisella BPEL-notaatiolla kuvattu prosessi. [32]

ne poikkeukset, jotka on määritelty web-sovelluspalvelun rajapinnassa. Tällöin tavoittamattoman palvelun kaltainen yleinen järjestelmävirhe jää havaitsematta. Poikkeus päätyy prosessitason catchAll-elementtiin, joka sisältää vain yhden toimintamallin. Sama toiminnallisuus suoritetaan kaikille poikkeustyypille. Useissa tilanteissa tarkoituksenmukaisempaa olisi mahdollisuus toimia eri tavoin sen mukaan, mikä kutsutuista palveluista oli tavoittamattomissa.

- BPEL ei tue muuttujien dynaamista luontia tai tuhoamista. Jokainen muuttuja on määriteltävä prosessitasolla, eikä paikallisten muuttujien luominen aktiviteettiryhmien sisälle ole mahdollista. Kompleksisen tietomallin sisältämän sanoman muuntaminen toisenlaiseen tietomalliin on BPEL:n tarjoamilla aktiviteeteilla työlästä.

Näiden rajoitusten takia on usein tarpeellista toteuttaa koostekohtaisia, sisäiseen käyttöön tarkoitettuja edustapalveluja, jotka toimivat kääreinä varsinaisille, toiminnallisuuden toteuttaville palveluille tai legacy-järjestelmille. Edustapalvelut on usein toteutettava koostepalvelun yhteyteen, sillä mahdollisuudet muokata alkuperäisen palvelun ominaisuuksia on monesti koostepalvelun toteuttajatahon ulottumattomissa. Edustapalvelu sisältää yleensä seuraavat toiminnot:

- varsinaisen palvelun kutsuminen
- datatransformaatio ennen ja jälkeen kutsun
- poikkeusten tyypittäminen



Kuva 5: Legacy-järjestelmän edustapalvelu.

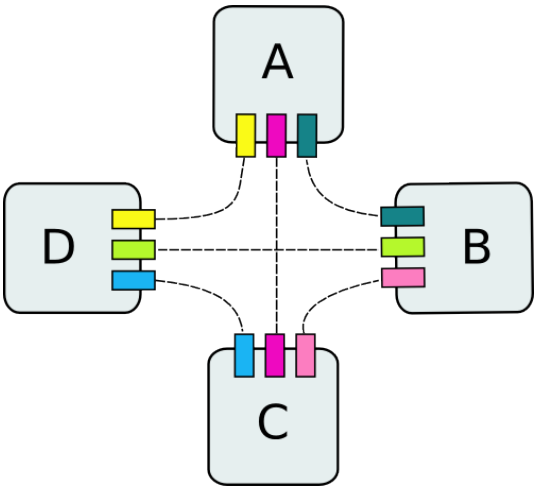
Edustapalvelu mahdollistaa legacy-järjestelmien toiminnallisuuksien hyödyntämisen palvelusuuntautuneen arkkitehtuurin mukaisesti. Edustapalvelu voidaan julkaista WSDL-rajapinnan avulla. Edustapalvelu tarjoaa käytännössä saman toiminnallisuuden kuin sen kutsuma, alkuperäinen palvelu tai järjestelmä, mutta uudelleenkäytettävänä web-sovelluspalveluna. Kuvassa 5 on esitetty, kuinka legacy-järjestelmän toiminnallisuus voidaan julkaista edustapalveluna toimivan web-sovelluspalvelun avulla.

2.6.2 Palveluväylä

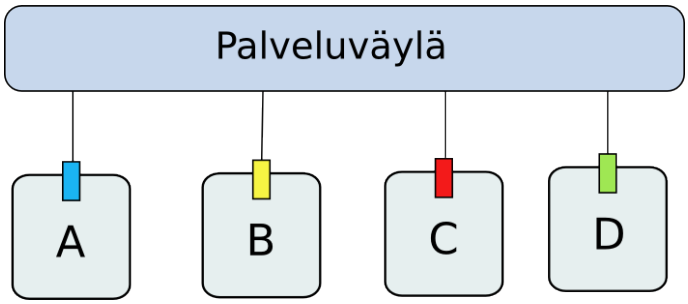
Palveluväylä (Enterprise Service Bus, ESB), on konsepti, jonka tavoitteena on luoda palvelusuuntautuneen arkkitehtuurin tarpeita vastaava infrastruktuuri.[19] Web-sovelluspalvelujen avulla pystytään yhdistämään erillisiä tietojärjestelmiä palvelusuuntautuneen arkkitehtuurin mukaisesti, mutta laajemmissa toteutuksissa niiden käyttö saattaa johtaa arkkitehtuuriltaan hauraaseen pisteestä pisteeseen -yhteyksien verkoksi. Lisäksi palveluja käyttävät sovellukset saattavat vaatia useita erilaisia sovittimia pystyäkseen viestimään erilaisten palvelujen kanssa. Tämä saattaa johtaa suuren työmäärään sovellusten ylläpidossa. Kuvassa 6 on havainnollistettu, kuinka vain neljän tietojärjestelmän yhdistäminen suorilla yhteyksillä johtaa kuuden yhteyden ja mahdollisesti jopa kahdentoista sovittimen kehittämiseen ja ylläpitoon.

Palveluväylälle ei ole tarkkaa määritelmää, vaan se on hyväksi havaittu käytäntö palvelusuuntautuneen arkkitehtuurin vaatiman infrastruktuurin toteuttamiseksi. [19] Se toimii palvelujen yhteisenä viestintäkanavana, joka huolehtii sanomien reitityksestä sekä protokollamuunnoksista. Tällöin jokainen palvelu tai palveluita hyödyntävä sovellus tarvitsee vain yhden protokollasovittimen, joka yhdistää sen palveluväylään. Kuvassa 7 on kuvattu neljän tietojärjestelmän yhdistävä palveluväylä.

Erilaisista palveluväylätoteutuksista on tunnistettavissa joitakin kaikille yhteisiä ja keskeisiä ominaisuuksia [19]:



Kuva 6: Pisteestä pisteeseen -yhteyksien verkko.



Kuva 7: Palveluväylän yhdistämät tietojärjestelmät.

- tarjoaa palvelutoteutuksille suoritusympäristön, joka on luotettava, vikasietoinen sekä tietoturvallinen
- palveluväylän palvelut ovat kaikkien palvelujen ja sovellusten käytettävissä ilman lisäohjelmointia
- on skaalattavissa mielivaltaisen kokoisen sovellusjoukon palvelemiseen
- tukee sanomien sisältöön perustuvaa reititystä sekä sanomien muunnosta
- mahdollistaa palvelujen orkestroinnin sekä niiden integroinnin ulkoisiin prosessimoottoreihin
- sisältää tuen tapahtumien ja ilmoitusten hallintaan

Palveluväylän tavoitteena on siis virtualisoida organisaation resurssit niin, että liiketoimintalogiikkaa voidaan kehittää ja ylläpitää verkkoinfrastruktuurista ja palvelujen sijainnista riippumattomasti. [8, s.39] Tämä mahdollistaa esimerkiksi palvelutoteutuksen uusimisen ja siirtämisen uuteen suoritusympäristöön ilman, että sitä käyttäviin asiakassovelluksiin tarvitaan muutoksia. Palveluväylä on vastuussa palvelujen paikallistamisesta ja sanomien reitittämisestä. Lisäksi palveluväylä voi tarjota

toimintoja palvelujen suorittamiseen, hallinnointiin ja oikeuksienhallintaan. Palveluväyläympäristöön toteutettavista sovelluksista, jotka muuntavat tai reitittävät sanomia tai toteuttavat muita lisätoimintoja, voidaan käyttää nimitystä *välityssovellus*. Välityssovellus on palveluista riippumaton sovellus, joka tyypillisesti sisältää esimerkiksi monipuolisia datatransformaatioon käytettäviä toiminnallisuuksia. Tämän vuoksi välityssovellukset sopivat hyvin esimerkiksi edustapalvelujen toteuttamiseen.

Kaupallisia toteutuksia palveluväylästä ovat muun muassa IBM WebSphere ESB [36], Microsoft BizTalk Server [37] sekä Oracle Service Bus [38]. Merkittäviä avoimen lähdekoodin toteutuksia ovat esimerkiksi Apache Synapse [39] sekä JBoss ESB [40].

2.7 Yhteenveto

Välikerrosohjelmistot tarjoavat sanomanvälitysinfrastruktuurin ja yhdenmukaisen ohjelmointirajapinnan, joiden avulla toteutustekniikoiltaan erilaiset tietojärjestelmät pystytään integroimaan. Niiden tarjoamien toimintojen käyttäminen prosessinohjausjärjestelmässä voi kuitenkin olla hankalaa teknisistä ja hallinnollisista syistä. Palvelusuuntautunut arkkitehtuuri ja web-sovelluspalvelut mahdollistavat tietojärjestelmien integroinnin yli organisaatiorajojen. Niitä käyttämällä pystytään yhdistämään eri tekniikoilla toteutettuja ja eri tahojen hallitsemia tietojärjestelmiä, joiden ainoa yhdistävä viestintäkanava on Internet. Palveluiden koostaminen on palvelusuuntautuneen arkkitehtuurin sadon korjaamista - se tuo esiin uudelleenkäytettävien palvelujen tuoman lisäarvon. Palvelukoosteita koordinoidaan tyypillisesti käyttäen orkestrointi-koordinointiparadigmaa, jossa yksi koordinaattori vastaa palvelujen kutsumisesta ja koosteen suorituslogiikasta. Koostepalvelun toteuttamisen mahdollisuus riippuu pitkälti palvelujen ominaisuuksista ja niiden onnistuneesta määrittelystä. Koostepalvelun orkestrointi voidaan toteuttaa käyttämällä BPEL-kieltä tai palveluväylätoteutusten välityssovelluksia.

3 Koostepalvelun transaktionhallinta

Tässä luvussa käsitellään hajautetun transaktion toteuttamista käyttäen keskitettyä transaktiokoordinointia sekä vaihtoehtoisia toteutustapoja silloin, kun transaktiokoordinoinnin käyttäminen ei ole mahdollista. Ensin esitellään transaktion, ja hajautetun transaktion ominaisuudet ja edellytykset. Tämän jälkeen tutustutaan kaksivaiheiseen vahvistamiskäytäntöön, joka on yleisesti käytetty hajautetun transaktion koordinointitapa. Kaksivaiheisen vahvistamiskäytännön käyttämisen mahdollisuus riippuu pitkälti osatransaktioiden ominaisuuksista, joten yleisiä palvelun toteutusteknologioita käsitellään siinä valossa, mitkä ovat niiden valmiudet osallistua kaksivaiheiseen vahvistamiskäytäntöön. Luvun lopuksi esitellään tapoja, joilla koostepalvelun transaktionaalisuus voidaan toteuttaa ilman transaktiokoordinointia.

3.1 Hajautettu transaktio

Jotta koostepalvelun kokonaistransaktionaalisuus toteutuu, sen automatisoinnissa on joko pystyttävä hyödyntämään palvelujen transaktiotoimintoja tai muulla tavalla huomioitava transaktionaalisuuden asettamat vaatimukset. Palvelujen transaktiotoimintojen hyödyntäminen vaatii, että järjestelmillä on yhteinen keino hajautetun transaktion hallintaan. Jos sitä ei ole, prosessin automatisointi on suunniteltava niin, että transaktionaalisuus kuitenkin toteutuu.

Hajautettu, eli globaali transaktio on transaktio, joka noutaa tai käsittelee useassa eri järjestelmässä sijaitsevaa tietoa käyttäen hyväksi paikallisen järjestelmän transaktioita [29, s.797]. Hajautetun transaktion suorittamiseksi sovellus kutsuu tietojärjestelmien transaktioproseduureja tai sovelluksia, jotka puolestaan itse toimivat hajautetun transaktion toteuttajina [13, s.781].

Alitransaktioiden voidaan olettaa noudattavan omassa järjestelmässään ACID-transaktioperiaatteita [13, s.24]. ACID on lyhenne sanoista atomicity, consistency, isolation, durability, suomeksi jakamattomuus, ristiriidattomuus, eristyvyys ja pysyvyys. Ne ovat tietokantajärjestelmien noudattamia periaatteita, jotka takaavat järjestelmän tietojen ehyyden kaikissa tilanteissa.

- jakamattomuus: joko kaikki transaktioon sisältyvät tehtävät suoritetaan, tai mitään niistä ei suoriteta
- ristiriidattomuus: järjestelmiin syötetään vain oikeita tietoja
- eristyvyys: keskeneräisen tai kumotun transaktion tulokset eivät ole näkyvissä muille transaktioille, eristäen täten niiden suorituksen toisistaan
- pysyvyys: suoritetun transaktion tulokset ovat pysyviä, eikä niitä voi kumota kuin kompensoivalla transaktiolla

ACID-määreiden tarvetta kuvaamaan voidaan käyttää esimerkkiä yksinkertaisesta transaktiosta, jonka tarkoituksena on siirtää 50 euroa tililtä A tilille B [29, luku 15]. Transaktion suorittamiseksi tarvitaan kaksi operaatiota:

- lue(X), joka lukee tietueen X arvon tietokannasta paikalliseen muistiin
- kirjoita(X), joka kirjoittaa paikallisen muuttujan X arvon paikallisesta muistista tietokantaan

Näillä operaatioilla suoritettuna transaktio toteutetaan seuraavasti:

```
lue(A);
A = A - 50;
kirjoita(A);
lue(B);
B = B + 50;
kirjoita(B);
```

Transaktion ristiriidattomuus toteutuu tässä transaktiossa sillä, että siirrettävän summan suuruus ei muutu transaktion suorittamisen aikana. Ilman tätä vaatimusta rahaa joko hävitettäisiin tai luotaisiin tyhjästä transaktion aikana. Ristiriidattomuus voidaan tässä tapauksessa todeta sillä, että tilien A ja B yhteenlaskettu saldo on sama ennen transaktion suorittamista ja sen suorittamisen jälkeen.

Jos transaktion suorittaminen syystä tai toisesta keskeytyisi kirjoita(A)-operaation jälkeen mutta ennen kirjoita(B)-operaatiota, tietokanta jäisi ristiriitaiseen tilaan, sillä rahasumma olisi vähennetty tililtä A, mutta sitä ei ole lisätty tilille B. Tilien yhteenlaskettu saldo ei siis olisi sama kuin lähtötilanteessa, ja rahat olisi hukattu. Tästä syystä transaktion tulee olla jakamaton: joko kaikki operaatiot suoritetaan tai mitään niistä ei suoriteta. Käytännössä tämä tapahtuu niin, että tietokantajärjestelmä pitää pysyvässä muistissaan tietueen alkuperäisen arvon niin kauan että transaktio on suoritettu loppuun asti. Jos transaktion suorittaminen jää puolitiehen, tietokantajärjestelmä palauttaa tietueeseen alkuperäisen arvon.

Pysyvyyden takaaminen edellyttää, että kaikki transaktion tekemät muutokset tallennetaan pysyvään muistiin ennen transaktion päättymistä. Tällöin muutokset pysyvät voimassa, vaikka transaktiota suorittava sovellus tai tietokantajärjestelmä kaatuisi transaktion suorittamisen jälkeen.

Vaikka transaktion ristiriidattomuudesta ja jakamattomuudesta olisi varmistuttu jokaisen yksittäisen transaktion tasolla, usean transaktion suorittaminen samanaikaisesti voi johtaa ei-toivottuihin lopputuloksiin, jos transaktiot pystyvät lukemaan toisen transaktion osittaisia tuloksia. Tämän vuoksi transaktioiden suorittaminen on eristettävä toisistaan. Transaktion eristyvyys voidaan toteuttaa suorittamalla transaktiot peräkkäin, jolloin transaktion suorittaminen ei ala, ennenkuin edellisen transaktion suorittaminen on päättynyt. Tästä kuitenkin seuraa merkittävää haittaa järjestelmän suorituskyvylle ja vasteajalle, kun jokainen transaktio joutuu odottamaan suoritusvuoroaan.

Suorituskyvyn parantamiseksi tietokantajärjestelmissä käytetään usein menetelmiä, jotka mahdollistavat transaktioiden yhdenaikaisesta suorittamisen niin, että eristyvyysehto täyttyy. Yleisimmin käytetty menetelmä on lukita osia tietokannasta vain yhden transaktion käyttöön. Lukitseminen koskee sekä luku- että kirjoitusoperaatioita. Lukitseminen kuitenkin voi johtaa tilanteisiin, jossa kahden tai useamman transaktioiden suorittaminen on riippuvainen saman lukittavan osion käytöstä, eikä

kumpikaan pysty etenemään. Tälläistä tilannetta kutsutaan lukkiumaksi (deadlock) [29, s.639].

Alitransaktiot tarjoavat hajautetun transaktion toteuttajalle abstrahoituja toimintoja. ACID-määreiden paikallinen toteutuminen jää alitransaktion vastuulle, ja hajautetun transaktion toteuttaja voi keskittyä hajautetun transaktion logiikan toteuttamiseen. Hajautettua transaktiota suorittavan sovelluksen vastuulle jää kuitenkin huolehtia hajautetun transaktion yhdistämän, integroidun järjestelmän eheydestä. Tämä edellyttää globaalin jakamattomuuden ja eristyvyyden takaamista. [13, s.784]

- Hajautetun transaktion jakamattomuus edellyttää, että jokainen alitransaktio on tahollaan jakamaton sekä että kaikki alitransaktiot vahvistetaan tai perutaan, kaikki-tai-ei-mitään -periaatteen mukaisesti.

- Eristyvyys edellyttää, että hajautetun transaktion kaikki alitransaktiot noudattavat eristyvyysperiaatetta, mutta myös sitä, että hajautettu transaktio on eristetty kaikista muista hajautetuista transaktioista. Tämä edellyttää globaalin sarjallistuvuuden toteutumista. Globaali sarjallistuvuuden edellytyksenä on, että jos kaksi hajautettua transaktiota, T1 ja T2, suoritetaan samanaikaisesti, kaikilla palvelimilla T1:n kaikkien alitransaktioiden suoritus näyttää tapahtuneen ennen T2:n alitransaktioita, tai kaikki T2:n alitransaktiot on suoritettu ennen T1:n alitransaktioita.

Globaalin jakamattomuuden ja eristyvyyden toteutuminen riittää takaamaan, että yhtäaikaisesti suoritetuilla hajautetuilla transaktioilla on sama vaikutus, kuin jos ne olisi suoritettu peräjälkeen jossakin järjestyksessä.

Jokaisella järjestelmällä on oma paikallinen transaktionhallitsin, joka varmistaa että paikallista tietoa käsitellään ACID-transaktioperiaatteiden mukaisesti. Lisäksi sen on ylläpidettävä järjestelmälokia virhetilasta palautumisen mahdollistamiseksi, sekä pystyä osallistumaan samassa lokaatiossa tapahtuvien transaktioiden yhdenaikaisuuden hallintaan.

Transaktiokoordinaattori puolestaan koordinoi kaikkia aloittamiaan, muissa järjestelmissä sijaitsevia alitransaktioita. Koordinaattorin tehtäviin kuuluu siis transaktion suorittamisen aloittaminen, transaktion jakaminen alitransaktioihin ja niiden jakaminen suoritettavaksi muihin järjestelmiin, sekä transaktion lopettamisen koordinointi, joka johtaa joko transaktion vahvistamiseen tai kumoamiseen.

Hajautetulle transaktionhallinnalle tyypillisiä ongelmatilanteita ovat: [29]

- transaktioon osallistuvien alijärjestelmien viat

- sanomien hukkuminen

- viestintäkanavan häiriöt

- tietoverkon jakautuminen

Yleisesti ottaen hajautettujen järjestelmien ongelmat voidaan nähdä liittyvän järjestelmien väliseen sanomavälitykseen. Vaikka käytössä olisikin TCP/IP-protokollan

kaltainen luotettavaa tiedonsiirtoprotokolla, sanoman hukkuminen on yhä mahdollista jos järjestelmät eivät ole suoraan yhteydessä toisiinsa, vaan sanoma reititetään usean verkkopisteen kautta. Joissain tilanteissa häiriöt tietoverkoissa saattavat johtaa siihen, että kahden järjestelmän välille ei löydy viestintäkanavaa, jolloin verkko on jakautunut erillisiin osioihin.

3.2 Kaksivaiheinen vahvistamiskäytäntö

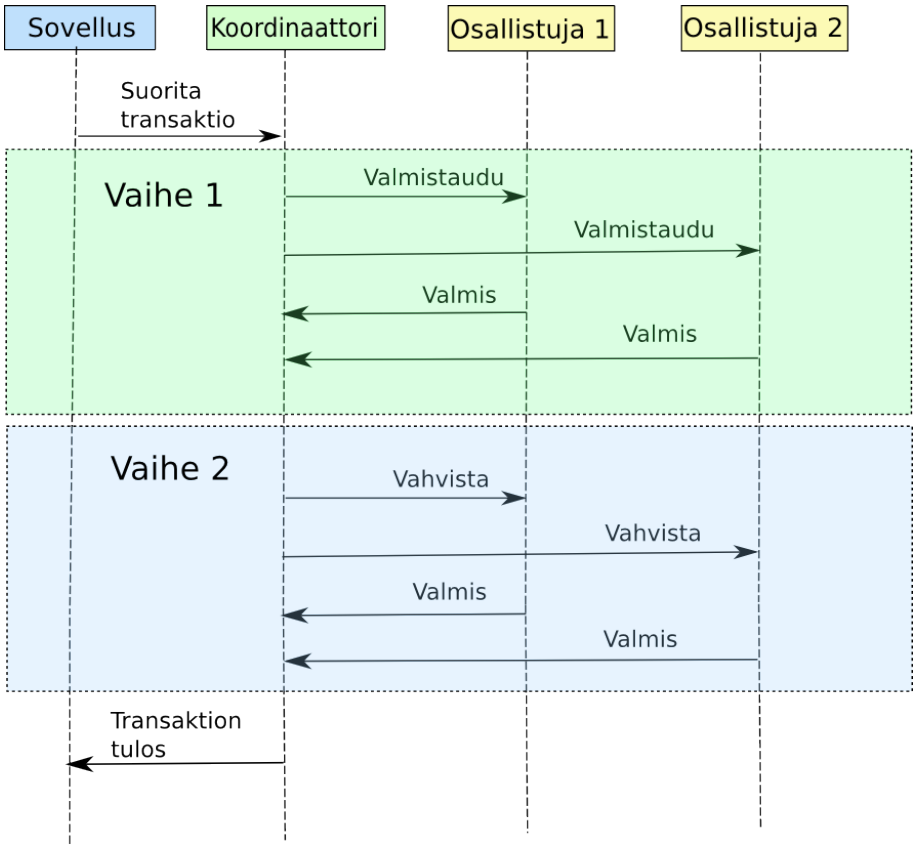
Transaktiokoordinaattorin on pystyttävä takaamaan hajautetun transaktion jakamattomuus. Tämä tarkoittaa, että hajautetun transaktion kaikkien alitransaktioiden on päätyttävä joko vahvistamiseen tai keskeytymiseen, mutta ei tilanteeseen, jossa osa transaktioista on vahvistettu ja osa keskeytetty. Jakamattomuuden toteuttamiseksi koordinaattorin on käytettävä vahvistamiskäytäntöä. Yleisimmin käytetty hajautetun transaktion koordinointitapa on kaksivaiheinen vahvistamiskäytäntö (two-phase commit protocol, 2PC). [13, s.1008]

Kuvassa 8 on esitetty kaksivaiheisen vahvistamiskäytännön kulku. Transaktion koordinaattori lähettää *valmistautumissanoman* (prepare message) koordinaatioon osallistuville transaktioille. Valmistautumissanoman tarkoituksena on selvittää, onko osallistuja halukas vahvistamaan transaktionsa. Tämä tarkoittaa, että osallistuja varmistaa, että transaktioon tarvittavat resurssit ovat käytettävissä, ja onko niiden tila sellainen, joka mahdollistaa transaktion suorittamisen ilman järjestelmän eheysrajoitteiden rikkomista. Jos osallistuja on halukas vahvistamaan transaktionsa, sen on valmistautumissanoman vastaanottaessaan tallennettava transaktionsa päivitystiedot pysyvään muistiin. Tallennus takaa sen, että osallistuja pystyy vahvistamaan transaktion, vaikka osallistujan järjestelmä kaatuisi vastattuaan myönteisesti valmistautumissanomaan.

Osallistuja vastaa valmistautumissanomaan äänestyssanomalla (vote message). Äänestyssanoman sisältö voi olla joko *valmis*, jos osallistuja on valmis vahvistamaan transaktion, tai *keskeyttää*, jos osallistuja on keskeyttänyt transaktionsa suorittamisen tai on muuten kykenemätön vahvistamaan sen. Äänestyssanoma on sitova, sillä koordinaattori käyttää sitä päättääkseen vahvistetaanko kokonaistransaktio.

Myönteisen äänestyssanoman lähetettyään osallistuja on *epävarmuusjaksossa*, sillä se ei tiedä tuleeko sen transaktio vahvistaa vai ei, ennen kuin koordinaattori on tehnyt ratkaisunsa. Tämän odotuksen aikana osallistujan on pidettävä alitransaktion resursseja lukittuina, jotta eheysrajoitteiden noudattamisesta voidaan varmistua. Jos osallistuja puolestaan lähettää kielteisen äänestyssanoman, alitransaktio perutaan välittömästi ja resurssit vapautetaan. Osallistujien äänestyssanomien lähettäminen päättää kaksivaiheisen vahvistamiskäytännön ensimmäisen vaiheen.

Koordinaattorin vastaanotettua kaikki äänestyssanomat se tekee päätöksen kokonaistransaktion vahvistamisesta. Jos kaikki äänestyssanomat ovat tyyppiä *valmis*, transaktio voidaan vahvistaa. Koordinaattori tallentaa pysyvään muistiin vahvistamispäätöksen ja sen jälkeen lähettää *vahvistussanoman* (commit message) kaikille osallistujille. Jälleen, vahvistamispäätöksen tallentaminen pysyvään muistiin on tärkeää sen tapauksen varalta, että koordinaattorin järjestelmä kaatuu vahvistamissanoman lähettämisen jälkeen. Tällöin oltaisiin tilanteessa, jossa alitransaktiot olisi



Kuva 8: Kaksivaiheinen vahvistamiskäytäntö.

vahvistettu, mutta kokonaistransaktion suorittamisesta ei olisi tietoa.

Osallistujan vastaanottaessa vahvistamissanoman se vahvistaa alitransaktion ja vapauttaa paikallisen järjestelmän resurssit. Tämän jälkeen osallistuja vastaa koordinaattorille *valmis*-sanomalla (done message). Koordinaattorin vastaanotettua kaikkien osallistujen valmis-sanomat se tallentaan pysyvään muistiin transaktion suoritustiedot ja pyyhkii ne ei-pysyvästä muistista, jonka jälkeen transaktion suorittaminen on päätöksessä. Jos yksikin koordinaattorin vastaanottamista äänestyssanomista on kielteinen, se lähettää kaikille osallistujille keskeytyssanoman. Keskeytyssanoma lopettaa osallistujien epävarmuusjakson.

Kaksivaiheisen vahvistamiskäytännön asettamat vaatimukset ovat ristiriidassa palvelukeskeisen arkkitehtuurin kanssa, jossa komponentit ovat löyhästi kytkettyjä ja sijaitsevat heterogeenisessa ympäristössä. Lisäksi resurssien lukitseminen eristyvyyden saavuttamiseksi on haastavaa erityisesti pitkäkestoisissa liiketoimintaprosesseissa. Kaksivaiheinen vahvistamiskäytäntö ei ole käyttökelpoinen pitkäkestoisissa transaktioissa seuraavista syistä [9]:

- Transaktioon osallistuvat palvelut eivät aina tue transaktionaalista käyttäytymistä. Keskitetyn koordinaattorin käyttäminen ei ole välttämättä mahdollista teknisten tai hallinnollisten rajoitteiden vuoksi, varsinkaan silloin, kun palvelut sijaitsevat eri organisaatioissa, ja transaktion hallintaa ei haluta luovuttaa organisaation ulkopuolelle.

- Eristyvyyden saavuttamiseksi tehtävä resurssilukitus ei sovellu pitkäkestoisiin prosesseihin sen aiheuttamien suorituskykyongelmien vuoksi.
- Kaksivaiheisen vahvistamiskäytännön implisiittinen kumoamistoiminto ei välttämättä sovellu palvelukeskeisen arkkitehtuurin mukaisiin toteutuksiin, joissa sekä toiminnon vahvistaminen että kumoaminen vaatii molemminpuolisen hyväksynnän.
- Hyvin suunniteltu palvelu tarjoaa monivaiheista toiminnallisuutta, joka muodostuu useasta toiminnosta ja komponenttien välisestä yhteistyöstä. Yksittäinen kumoamistoiminto ei välttämättä ole riittävä kumoamaan palvelukutsun seurauksia.
- Kumoamiseen vaaditut toiminnot eivät aina riipu pelkästään vain edellisestä palvelukutsusta, vaan myös kutsuvan prosessin tilasta. Löyhästi kytketyt palvelut eivät jaa tilatietoa, joten ne eivät aina pysty kumoamaan toistensa toimintoja.

3.3 Palvelutoteutusten transaktiotuki

Hajautetun transaktion hallinta käyttäen transaktiokoordinointia vaatii, että transaktioon sisältyvät palvelut pystyvät osallistumaan koordinointiprotokollan käyttöön. Tässä aliluvussa tarkastellaan joidenkin yleisesti palvelujen toteutukseen, tai niiden tukemiseen, käytettävien tekniikoiden kykyä osallistua hajautetun transaktion hallintaan.

3.3.1 Web-sovelluspalvelut

WS-Transaction on web-sovelluspalveluissa käytettävä standardoitu transaktionhallintatapa. Se muodostuu kolmesta erillisestä kokonaisuudesta: WS-Coordination (WS-C), WS-AtomicTransaction(WS-AT) ja WS-BusinessActivity (WS-BA). [41]

WS-Coordination määrittelee yleisen tavan, jolla hajautetun järjestelmän löyhästi kytketyt palvelut pystyvät koordinoituun yhteistoimintaan. Se ei määrittele varsinaista transaktioprotokollaa tai vahvistamiskäytäntöä, vaan toimii sellaisten mahdollistajana. WS-C määrittelee kahden tyyppisiä toimijoita, koordinaattoreita ja osallistujia. Koordinaattori luo koordinaatioasiayhteystiedon (coordination context), jota koordinaation osallistujat käyttävät sanomakehyksenä. Asiayhteystieto sisältää varsinaisen protokollan sanomat. Koordinaattori koordinoi osallistujien toimintoja käytettävän protokollan mukaisesti. Osallistujat ovat web-sovelluspalveluja, joiden palveluja kutsutaan osana transaktiota.

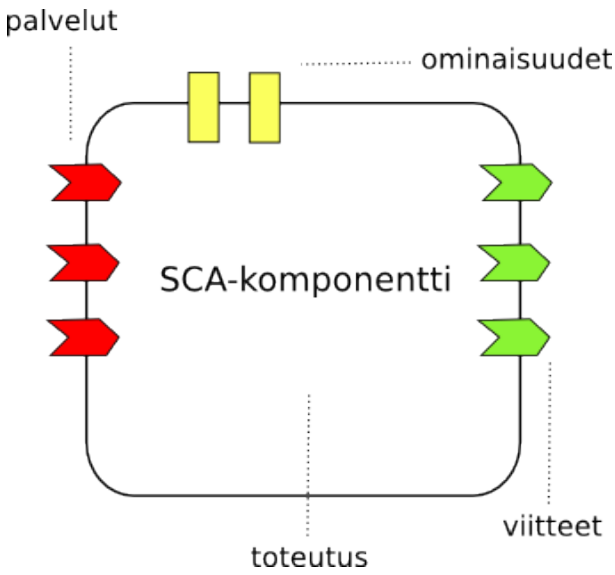
WS-C:n tarjoamissa puitteissa voidaan käyttää kahta hajautetun transaktion hallintaprotokollaa. WS-AtomicTransaction mahdollistaa web-sovelluspalvelujen osallistua hajautettuun transaktioon, tai koordinoida sitä, käyttäen kaksivaiheista vahvistamiskäytäntöä. WS-AT on tarkoitettu lyhytkestoisille transaktioille, sillä sen käyttö asettaa samoja rajoitteita kuin kaksivaiheisen vahvistamiskäytännön käyttäminen missä tahansa ympäristössä. WS-BusinessActivity on puolestaan tarkoitettu

pitkäkestoisille transaktioille, joissa lukkojen ylläpitäminen ei ole mahdollista. Sen sijaan WS-BA sisältää mahdollisuuden toteuttaa transaktioiden kompensoinnin.

3.3.2 Service Component Architecture

Service Component Architecture (SCA) on sovelluskehitysmalli ja -ympäristö, joka tukee palvelusuuntautuneen arkkitehtuurin mukaisten sovellusten kehitystä [25]. SCA:n tavoitteena on tukea mahdollisimman monia teknologioita, joita käytetään palveluiden toteuttamiseen tai niiden yhdistämiseen. SCA siis mahdollistaa palvelusuuntautuneiden liiketoimintasovellusten kehittämisen.

SCA-mallin mukaisen sovelluksen keskiössä on SCA-komponentti [26]. SCA-komponentti sisältää jonkin liiketoiminnallisen tehtävän toteutuksen. Se voi olla toteutettu yhdellä monista SCA-mallin tukemista ohjelmointikielistä tai muodostua muiden SCA-komponenttien yhdistelmästä. Komponentin liiketoimintatehtävän toteutuksessa tuetaan ohjelmointikieliä Java, Spring, BPEL, C, C++ ja Cobol. Komponentin toteuttama toiminnallisuus julkaistaan palveluna, ja toteutus voi olla riippuvainen muiden komponenttien tarjoamista palveluista. Toteutuksella voi olla myös ominaisuuksia, joita hallitaan komponettitasolla. Kuvassa 9 on havainnollistettu SCA-komponentin rakennetta. SCA-komponentin palvelurajapinnat mahdollistavat komponentin palvelujen käyttämisen osana toisten SCA-komponenttien toiminnallisuutta. Näihin toisiin komponentteihin yhdistämisen tieto sisältää SCA-komponentin viitteisiin.



Kuva 9: SCA-komponentti [42]

Komponentteja yhdistelemällä pystytään toteuttamaan palvelusuuntautuneen arkkitehtuurin mukaisesti uusia sovelluksia, käyttäen hyväksi palveluina julkaistuja liiketoimintatehtävien toteutuksia. Komponentin rajapinta voidaan määritellä joko Java-rajapintana tai WSDL-dokumenttina. Komponenttien kutsuun voidaan käyttää SOAP-, JMS-, EJB- tai JCA-sanomia, sekä toteutuskohtaisesti muitakin yleisesti käytössä olevia sanoma- tai etäkutsuteknologioita.

SCA-malli kehitettiin alunperin järjestelmätoimittajien Open SOA Collaboration -konsortiossa, johon kuuluu muun muassa IBM, Oracle, SAP, SUN ja Tibco. Määritelmien valmistuttua vuonna 2007 ne otettiin osaksi OASIS-standardeja.

3.3.3 Java Transaction API

Java Transaction API (JTA) määrittelee Java-pohjaisen standardirajapinnan hajautetun transaktion hallintaan. Sen tarkoituksena on määritellä paikalliset Java-rajapinnat, joita käytetään transaktion hallintaan hajautetussa, Java-pohjaisessa tietojärjestelmäympäristössä. [44] JTA sisältää kolme osaa:

- korkean abstraktiotason ohjelmointirajapinnan transaktionaalisen sovelluksen transaktiorajojen määrittelyyn
- Java-sovituksen X/Open XA -transaktionhallintaprotokollaan
- ohjelmointirajapinnan sovelluspalvelimen ylläpitämän sovelluksen transaktiorajojen määrittelyyn

Java Transaction API:a voidaan käyttää Java-sovellusten ja -resurssien hajautetun transaktion hallintaan. Näihin kuuluu muun muassa EJB-sovellukset sekä JDBC-yhdistimellä käytettävät tietokantaresurssit. X/Open XA on koordinaatioprotokolla, joka mahdollistaa kaksivaiheisen vahvistamiskäytännön käyttämisen sitä tukevissa tietojärjestelmissä [46].

3.3.4 IMS

IBM Information Management System (IMS) on transaktio- ja tietokantahallintaohjelma. [43]

IMS-sovelluksia on mahdollista kutsua IMS Connect -yhdyskäytävän kautta. Sen vastinpari, asiakassovellukseen toteutettava IMS Connector muodostaa yhteyden IMS Connectiin käyttämällä TCP/IP-protokollaa. IMS Connect tarjoaa yleiskäyttöisen rajapinnan IMS-yhdistimille sekä muille asiakassovelluksille. Se sallii useita yhdenaikaisia TCP/IP yhteyksiä IMS-transaktiopalvelimen transaktioihin ja komentoihin.

IMS Connect mahdollistaa IMS-sovellusten osallistumisen hajautettuihin transaktioihin kaksivaiheisella vahvistamiskäytännöllä. Se tukee myös asynkronista sanomavälitystä. Jos asiakassovellus on toteutettu käyttäen IMS Connectoria, asiakassovelluksen on mahdollista hakea kerrallaan yhden transaktiokutsun vastausviesti. Käyttäen suoraan IMS:n sisäistä OTMA -protokollaa on mahdollista noutaa myös kaikki sanomajonossa odottavat sanomat. Asiakassovellus on mahdollista toteuttaa myös ilman IMS Connectoria, millä tahansa ohjelmointikielellä, joka tukee TCP/IP-protokollaa. Tällöin asiakassovelluksen toteuttajan on ymmärrettävä IMS Connectin sekä IMS:n OTMA -protokollien toiminta.

Toinen mahdollinen yhteysmalli on IMS SOAP Gateway -yhdyskäytävä. Se mahdollistaa IMS-sovelluksen julkaisemisen SOAP/HTTP 1.1 ja WSDL 1.1 standardien

mukaisena web-sovelluspalveluna. SOAP Gatewayn käyttöönotto ei vaadi muutoksia varsinaiseen IMS-sovellukseen. SOAP Gateway ei tue monivaiheisia transaktioita, kuten kaksivaiheista vahvistamiskäytäntöä. [11]

3.3.5 CICS-transaktiopalvelin

CICS-transaktiopalvelimen [45] sovellusten toiminnallisuuksia voidaan palveluina käyttäen joko edustapalveluna toimivan web-sovelluspalvelun avulla tai sovittimen avulla. Web-sovelluspalvelu toteutetaan varsinaiselle CICS-transaktiopalvelimella. CICS-transaktiopalvelin voi myös osallistua sekä koordinoida hajautettuja transaktioita, joissa on osallisina WS-AT-protokollaa noudattavia web-sovelluspalveluja. Pitkäkestoisten transaktioiden hallintaan tarvittavat kompensoivat transaktiot voidaan toteuttaa web-sovelluspalveluina aivan kuten ne transaktiot, joita ne kompensoivat.

Sovitin puolestaan toteutetaan erilliselle sovelluspalvelimelle. Sovitin muuntaa saapuvan sanoman CICS-sovellukselle sopivaan sanomamuotoon. Sovittimia on saatavilla monille XML-pohjaisille sanomatyypeille sekä JSON-sanomaformaatille. Sovelluspalvelin ja CICS-palvelin yhdistetään toisiinsa soveltuvalla yhdistimellä, kuten CICS Transaction Gateway. Sovitin, joka käyttää Java EE Connector Architecturen (JCA) mukaista yhdistintä, voi koordinoida hajautettuja transaktioita, joissa on osallisena CICS-transaktiopalvelimen sovelluksia. [10]

3.3.6 Yhteenveto

Taulukossa 1 on vedetty yhteen edellä esiteltyjen palvelutoteutusten tuki hajautetun transaktion koordinaatiolle.

Taulukko 1: Palvelutoteutusten transaktiotuki

Toteutustekniikka	Kaksivaiheinen vahvistamiskäytäntö	Pitkäkestoiset transaktiot	X/Open XA
Web-sovelluspalvelut	kyllä	kyllä	ei
SCA	kyllä	ei	ei
JTA	ei	ei	kyllä
IMS	kyllä	kyllä	ei
CICS	kyllä	kyllä	ei

3.4 Pitkäkestoisten transaktioiden hallinta

Tiukan transaktionhallinnan toteuttaminen esimerkiksi käyttämällä kaksivaiheista vahvistamiskäytäntöä vaatii tiettyjen lähtökohtien täyttämistä:

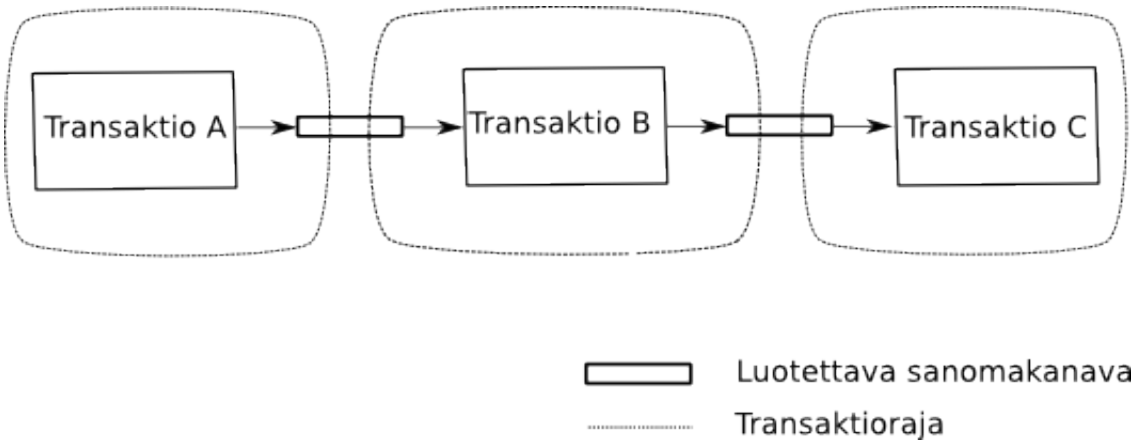
- transaktioiden on oltava luonteeltaan lyhytkestoisia

- palvelut ovat halukkaita luovuttamaan transaktioidensa koordinoinnin ulkopuoliselle koordinaattorille
- palveluiden transaktioiden koordinointi on teknisesti mahdollista

Tietokantojen lukkiumat lisääntyvät nopeasti transaktioiden keston pidentyessä. Tämän vuoksi pitkäkestoiset transaktiot eivät voi ylläpitää lukkoja hallitsemissaan resursseja. Transaktiot voivat ylläpitää lukkoja vain silloin, kun ne tekevät muutoksia tietokantaan. Tämä tarkoittaa sitä, että transaktion tekemät muutokset ovat muiden transaktioiden nähtävissä, ennen kuin se on vahvistettu lopullisesti [15]. Pitkäkestoisten transaktioiden kohdalla on siis hyväksyttävä lyhytkestoista transaktiota löyhemmät rajoitteet: transaktion eristyvyydestä joudutaan luopumaan, ja riski ristiriitatilanteista kasvaa.

Jotta hajautetun transaktion hallintaprotokollat voivat toimia, niillä on oltava oikeus hallita paikallisia transaktioita, eli määrätä milloin transaktio vahvistetaan tai milloin se perutaan. Ympäristössä, jossa palveluita hallitsevat erilliset organisaatiot, transaktionhallinnan myöntäminen ulkopuolisille ei välttämättä ole tarkoituksenmukaista.

Jos hajautetun transaktion hallinnan vaatimat ehdot eivät täyty, transaktio on pilkottava alitransaktioiksi, jotka suoritetaan peräjälkeen [13, s.788]. Tälläisessä ketjutetussa transaktiossa jokainen alitransaktio vahvistetaan vuorollaan ja sen tulokset tallennetaan pysyvään muistiin. Tällöin tietokannoissa ei jouduta ylläpitämään lukkoja koko ketjutetun transaktion keston ajan, vaan ainoastaan suoritusvuorossa oleva alitransaktio lukitsee käyttämänsä tietokantataulut ja vapauttaa ne välittömästi suorituksensa jälkeen. Jokainen alitransaktio hallitsee siis itse vahvistamistaan, eikä hallintaoikeuksia tarvitse luovuttaa ulkopuoliselle taholle. Kuvassa 10 on kuvattu ketjutettuja transaktioita transaktiorajoineen.



Kuva 10: Ketjutettuja transaktioita.

Alitransaktioiden välillä on käytettävä luotettavaa sanomavälitystä [29, s.844]. Luotettavalla sanomavälityksellä tarkoitetaan tässä järjestelmää, joka takaa että sille luovutettu sanoma toimitetaan vastaanottajalle kerran, ja vain kerran. Sekä sanoman vastaanottaja että sen lähettäjän täytyy tällöin toimia transaktionaalisesti.

Transaktionaalinen vastaanottaja vastaanottaa sanoman ilman, että se poistetaan sanomajärjestelmän jonosta. Vasta kun sanoma on onnistuneesti käsitelty, ja siihen on mahdollisesti luotu vastaus joka on onnistuneesti asetettu lähtevään sanomakanavaan, vastaanotettu sanoma poistetaan järjestelmän jonosta. Luotettavan sanomavälityksen käyttäminen varmistaa että vikatilanteessa tieto transaktioketjun tilasta säilyy, ja sovellus voi toteuttaa kompensaation.

Transaktion toteuttaminen alitransaktioita ketjuttamalla tuo esiin kuitenkin useita huomioonotettavia seikkoja [13] :

1. Jos alitransaktion suorittamisessa tapahtuu järjestelmän vikatilanne, sitä edeltäneiden alitransaktioiden tuloksia ei menetetä, sillä ne on jo vahvistettu ja niiden tulokset tallennettu pysyvään muistiin.

2. Ketjuttamalla toteutettu hajautettu transaktio ei täytä jakamattomuuden transaktionaalisuusperiaatetta.

3. Alitransaktiot ovat hajautetun transaktion osatehtäviä, joten niiden on yleensä pystyttävä välittämään sanomia keskenään. Alitransaktion suorittamiseen käytettävät arvot saattavat esimerkiksi olla riippuvaisia edeltävän alitransaktion tuloksista. Alitransaktioiden välisessä sanomavälityksessä on otettava huomioon sanomien säilyminen vikatilanteissa.

4. Alitransaktioita ketjuttamalla toteutetulla hajautetulla transaktiolla ei ole eristyvyyttä, sillä resurssien tilat voivat muuttua alitransaktioiden välillä.

Virheeseen päätyneen transaktioketjun suoritusta ei siis voida kumota käyttämällä esimerkiksi tietokantajärjestelmien kumoamistoiminnallisuutta, sillä eristyvyyden puuttuessa ei voida olettaa että tietokantojen tietueet ovat samassa tilassa kuin välittömästi kunkin alitransaktion suorittamisen päätyttyä. Transaktioketjun suorittavan sovelluksen onkin toteutettava toiminnallisuus, jolla ketjun suorittaminen voidaan hallitusti viedä loppuun tai tehdyt suoritukset kumota. Tämä saadaan aikaiseksi kompensoimalla jo loppuun vietyjen alitransaktioiden tulokset.

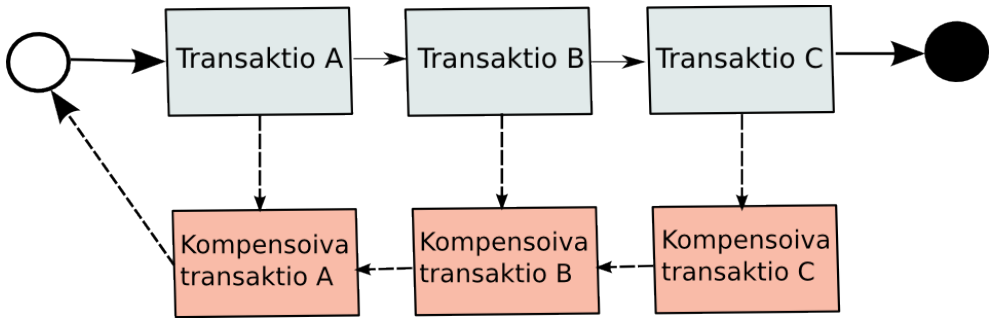
Kompensaatio on toiminto, joka loogisesti kumoaa suoritetun toiminnon tuloksen. Kompensoiva transaktio ei siis välttämättä palauta tietokantaa transaktion suorittamista edeltävään tilaan, vaan suorittaa toimintoja, jotka loogisesti kumoavat alkuperäisen transaktion tulokset. Kompensaatio voi olla myös monimutkaisempi, liiketoimintalogiikkaa sisältävä komponentti, jonka suorittaminen kumoaa suoritetun tehtävän tulokset suorittamalla useita palvelu- ja transaktiokutsuja. Kompensoivia transaktioita voidaan käyttää:

- palauttamaan järjestelmän eheys tilanteessa, jossa hajautettu transaktio on keskeytynyt tavalla, joka estää transaktion normaalin päätökseenviennin

- kumoamaan hajautetun transaktion operaatiot tilanteessa, jossa transaktiokoordinaattori on saanut vain osalta kaksivaiheiseen vahvistamiskäytäntöön osallistuvilta tahoilta toisen vaiheen vastauksen, eikä transaktion jakamattomuudesta ole varmuutta

- hajautettuun transaktioon osallistuvan ei-transaktionaalisen resurssin tuloksen kumoamiseen transaktion kumoamistilanteessa

Transaktioketjua, jonka jokaiselle alitransaktiolle pystytään määrittelemään kompensaatiotransaktio, kutsutaan *saagaksi* [14] [13, s.791]. Kuvassa 11 on esitetty, kuinka saagan tulokset voidaan kumota, jos jokin ketjun transaktioista epäonnistuu. Saagan päätasolla ei saavuteta täyttä jakamattomuutta, sillä saagalla on näkyvyys toisiin, keskeneräisiin saagoihin.



Kuva 11: Saaga.

Käytännössä kompensaatio on suoritettavan liiketoimintaprosessin määritelmän tai koostepalvelun logiikan jatke. Koostepalvelun orkestroinnin on otettava huomioon myös virheet ja niiden asianmukainen käsittely. Tähän käytetään prosessinkuvauskieleen elementtejä, kuten kompensaatiorajoja (compensation scope), virheiden signalointimekanismeja sekä kompensointitehtävien määrittelyn. Orkestrointia käytettäessä prosessin tilatietoa voidaan käyttää hyväksi tilariippuvaisen kompensaation määrittelyssä. [9]

Kompensaation käyttäminen ei vaadi palveluilta varsinaisia transaktionaalisia kyvykkyyksiä, sillä suoritettujen kutsujen tulokset voidaan kumota soveltuvilla lisäkutsuilla. Saagaan sisältyvät transaktionaaliset palvelut eivät jää odottamaan vahvistamis- tai kumoamiskutsuja, jolloin resurssien pitkäaikaiselta lukitsemiselta vältytään. Tämän vuoksi saagat sopivat erityisen hyvin pitkäkestoisten transaktioiden toteuttamiseen ympäristössä, jossa varsinaisen hajautetun transaktion toteuttaminen on hankalaa.

Kompensaatio ei kuitenkaan ratkaise eristyvyyden puuttumisesta seuraavia ongelmia. Kun kahdella tai useammalla pitkäkestoisella transaktiolla on samasta resurssista erilainen tilatieto ja yksi transaktioista epäonnistuu, seuraa tilanne jossa suoritustaan jatkavilla transaktioilla on väärä tieto resurssin tilasta. Tämän ongelman ratkaisuksi on ehdotettu suoritusriippuvuustiedon rekisteröimistä pitkäkestoisille transaktioille. [27] Konseptia toteuttavia protokollia ei ole kuitenkaan vielä standardoitu, eikä niille siten löydy tukea palveluiden toteutustekniikoista.

3.5 Yhteenveto

Transaktionaalisuuden toteutuminen on tärkeää monissa liiketoiminnoissa. Transaktioille on määritelty ACID-periaatteet, joiden noudattaminen varmistaa transaktion

oikean toiminnan. Useat alitransaktiot voidaan koota osaksi hajautettua transaktiota. Niiden suorituksen hallitsemiseksi tarvitaan vahvistamiskäytöntöä. Kaksivaiheinen vahvistamiskäytäntö on yleisesti käytetty hajautetun transaktion koordinointitapa. Sen käyttäminen vaatii kuitenkin kaikkien alitransaktioiden tuen, joka voi olla heterogeenisessä tietojärjestelmäympäristössä mahdotonta. Lisäksi, pitkäkestoisissa tehtävissä hajautetun transaktioiden eristyvyydestä joudutaan luopumaan tietojärjestelmien suorituskyvyn turvaamiseksi. Kun hajautetun transaktion koordinointi ei ole mahdollista, se voidaan toteuttaa ketjuttamalla yksittäisiä transaktioita. Tällöin transaktioiden välillä on käytettävä luotettavaa sanomavälitystä. Jotta alitransaktioiden tulokset voidaan tarvittaessa kumota, niille on määriteltävä kompensoivia transaktioita. Transaktioketjua, jonka kaikilla alitransaktioilla on kompensoiva transaktio, kutsutaan saagaksi. Saaga toteuttaa transaktion ACID-periaatteet eristyvyyttä lukuunottamatta.

4 Koostepalvelujen suunnittelu

Tässä luvussa esitellään kaksi koostepalvelutoteutusta sekä selvitetään, kuinka ne toteuttavat transaktionaaliselle koostepalvelulle asetetut vaatimukset. Aluksi määritellään, mitä koostepalvelulla tarkoitetaan, ja mitä sen transaktionaalisuudelta odotetaan. Toteutusympäristö, eli käytettävät kehitys- ja suoritusympäristöt, esitellään. Koostepalvelujen lähtökohdat, käytettävissä olevat palvelut ja niiden toteutus käydään läpi. Tämän jälkeen arvioidaan saavuttivatko toteutukset niille asetetut vaatimukset koostepalvelun toiminnallisuuden ja transaktionaalisuuden suhteen.

4.1 Tavoitteet ja määritelmät

Tässä yhteydessä koostepalvelulla tarkoitetaan palvelua, joka:

- tarjoaa uutta toiminnallisuutta tekemällä useamman kuin yhden palvelukutsun
- omaa julkaistavissa olevan WSDL-rajapinnan
- on toteutettu noudattaen palvelusuuntautunutta arkkitehtuuria
- sisältää transaktionhallinnan

Koostepalvelun transaktionhallinta huolehtii, että koostepalvelun toteuttama hajautettu transaktio toimii koko integroidun järjestelmän laajuudella tarkoituksenmukaisella tavalla. Tähän sisältyy erityisesti integroidun järjestelmän eheyden takaaminen. Lisäksi transaktionhallinnalta edellytetään seuraavia ominaisuuksia:

- ei aiheuta palvelujen resurssien pitkäaikaista lukittumista
- kumoutumistilanteessa kumoaa tai kompensoi jo suoritettujen palvelukutsujen tulokset

Toteutettavat koostepalvelut toteuttavat prosessinohjausjärjestelmille tyypillisiä palveluprosesseja. Ensimmäiseksi esiteltävässä koostepalvelussa käsitellään organisaatioon saapuva hakemus, toisessa koostepalvelussa lasketaan hakemuksen täyttämisestä koituvien maksujen suuruus. Koostepalvelujen käytössä on erityyppisiä tietojärjestelmiä, jotka toteuttavat osan koostepalvelun tehtävistä, mutta osa tehtävistä on suoritettava manuaalisesti.

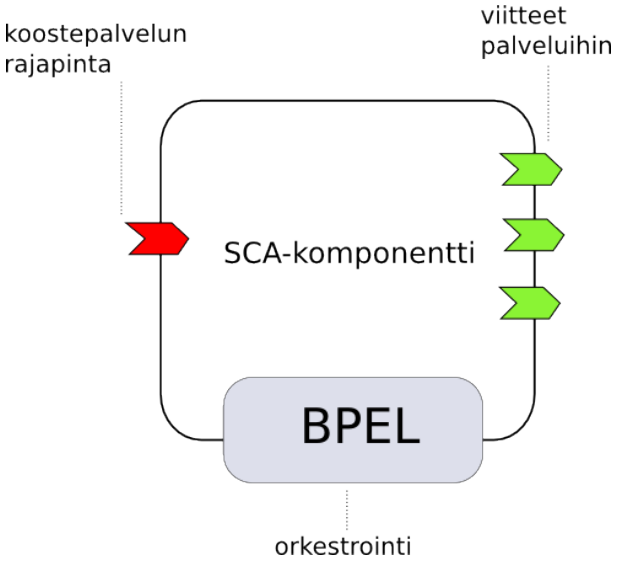
4.2 Toteutusympäristö

Koostepalvelujen toteutusympäristönä toimii IBM Business Process Manager Advanced (BPM). Toteutuksen kannalta sen tärkeimmät osiot ovat:

- prosessiserveri IBM Process Server
- palveluväylä IBM WebSphere Enterprise Service Bus

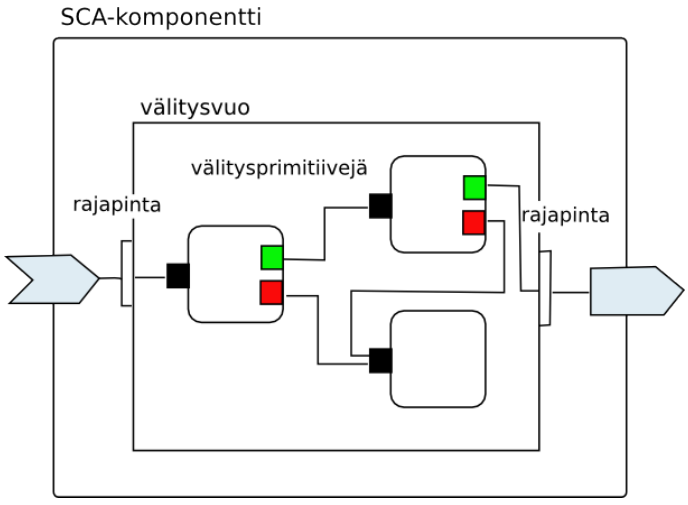
- kehitysympäristö IBM WebSphere Integration Designer

IBM Process Server pohjautuu sovelluspalvelimen arkkitehtuuriin, joka mahdollistaa muun muassa JavaEE-sovellusten suorittamisen [23]. Täten BPEL-orkestraatio voidaan toteuttaa JavaEE-sovelluksena SCA-komponenttiin. BPEL-kuvauksen suorittamiseksi IBM Process Server sisältää prosessimoottorin.



Kuva 12: BPEL-orkestrointitoteutus SCA-komponentissa. [23, s.46]

WebSphere Enterprise Service Bus toteuttaa palveluväylän infrastruktuurin. Se mahdollistaa myös välityssovellusten toteuttamisen. Välityssovellus toteutetaan välitysvuotyyppisenä SCA-komponenttina. Välitysvuo sisältää joukon välitysprimitiivejä, jotka toteuttavat sanomanvälitykseen ja -muunnokseen tarvittavia toimintoja. Välityssovellus voi täten muuttaa palvelukutsun formaattia, sisältöä tai kohdetta [24].



Kuva 13: Välitysvuo SCA-komponentissa. [24]

Integration Designer on Eclipse-sovelluskehitysympäristöön pohjautuva, erityisesti prosessinohjaus- ja järjestelmäintegraatiosovellusten kehittämiseen suunniteltu kehitysympäristö. Se mahdollistaa muun muassa BPEL-prosessikuvausten ja välitysmediaatioiden kehittämisen graafisen käyttöliittymän avulla, ja sisältää siis oman graafisen notaation BPEL-kielelle.

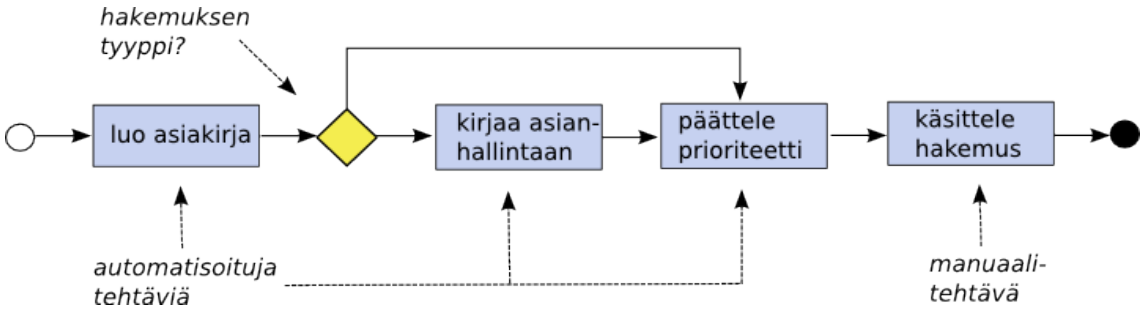
4.3 Koostepalvelu 1: Hakemuksen käsittely

Tässä aliluvussa kuvataan Hakemuksen käsittely -koostepalvelulle asetetut vaatimukset sekä esitellään koostepalvelun toteutus.

4.3.1 Palvelun kuvaus

Hakemuksen käsittelyyn sisältyy neljä erillistä tehtävää:

- käsittelyasiakirjan luominen
- hakemuskäsittelyn kirjaaminen asianhallintajärjestelmään
- hakemuksen käsittelyprioriteetin päättely
- hakemuksen käsittely



Kuva 14: Hakemuksen käsittely.

Kuvassa 14 on esitetty koostepalvelun tehtävien suoritusjärjestys. Hakemuskäsittelyn kirjaaminen asianhallintajärjestelmään on ehdollinen riippuen hakemuksen tyypistä. Kolmen ensiksi suoritettavaa tehtävää on automatisoitu ja niiden suorittaminen siis onnistuu tekemällä palvelukutsut niitä ylläpitäviin tietojärjestelmiin. Neljäs tehtävä, hakemuksen käsittely, suoritetaan manuaalityönä. Jos käsittely keskeytyy käsittelyasiakirjan luomisen jälkeen, luotu asiakirja on poistettava.

4.3.2 Käytettävissä olevat palvelut

Käsittelyasiakirjan luomiseen käytetään Asiakirjapalvelun web-sovelluspalvelua. Asiakirjapalvelun luoAsiakirja-toiminto luo asiakirjajärjestelmään uuden asiakirjan, joka toimii tallenteena käsittelyyn otetusta hakemuksesta. Palvelun kutsusanomassa välitetään tunnistetietojen lisäksi hakemuksen perusteella luotu HTML-dokumentti

Base64-muunnettuna [28] ASCII-merkkijonona. Palveluun ei ole toteutettu WS-Transaction-protokollan mukaista hajautetun transaktiokoordinoinnin tukea.

Hakemuskäsittelyn kirjaaminen suoritetaan kutsumalla tietokantasovelluksen tallennettua proseduuria. Tallennettu proseduuri on tietokantapalvelimelle tallennettu SQL-kysely [13, s.282]. Pyyntösanomassa välitetään hakemuksen yksilöivä tunnus sekä muita hakemuksen tietoja. Tiedot palvelun toiminnasta sekä vastaussanomista virhetilanteissa ovat puutteellisia.

Käsittelyprioriteetin päättely on prosessiserverille toteutettu SCA-arkkitehtuurin mukainen JavaEE-sovellus. Täten se pystyy osallistumaan koordinoituun transaktionhallintaan kaksivaiheisen vahvistamiskäytännön avulla.

Taulukossa 2 on vedetty yhteen käytössä olevat palvelut, niiden tarjoamat toiminnot, toteutustavat sekä kyvykkyys osallistua koordinoituun hajautetun transaktion hallintaan.

Taulukko 2: Hakemuksen käsittelyn tehtäviä toteuttavat palvelut

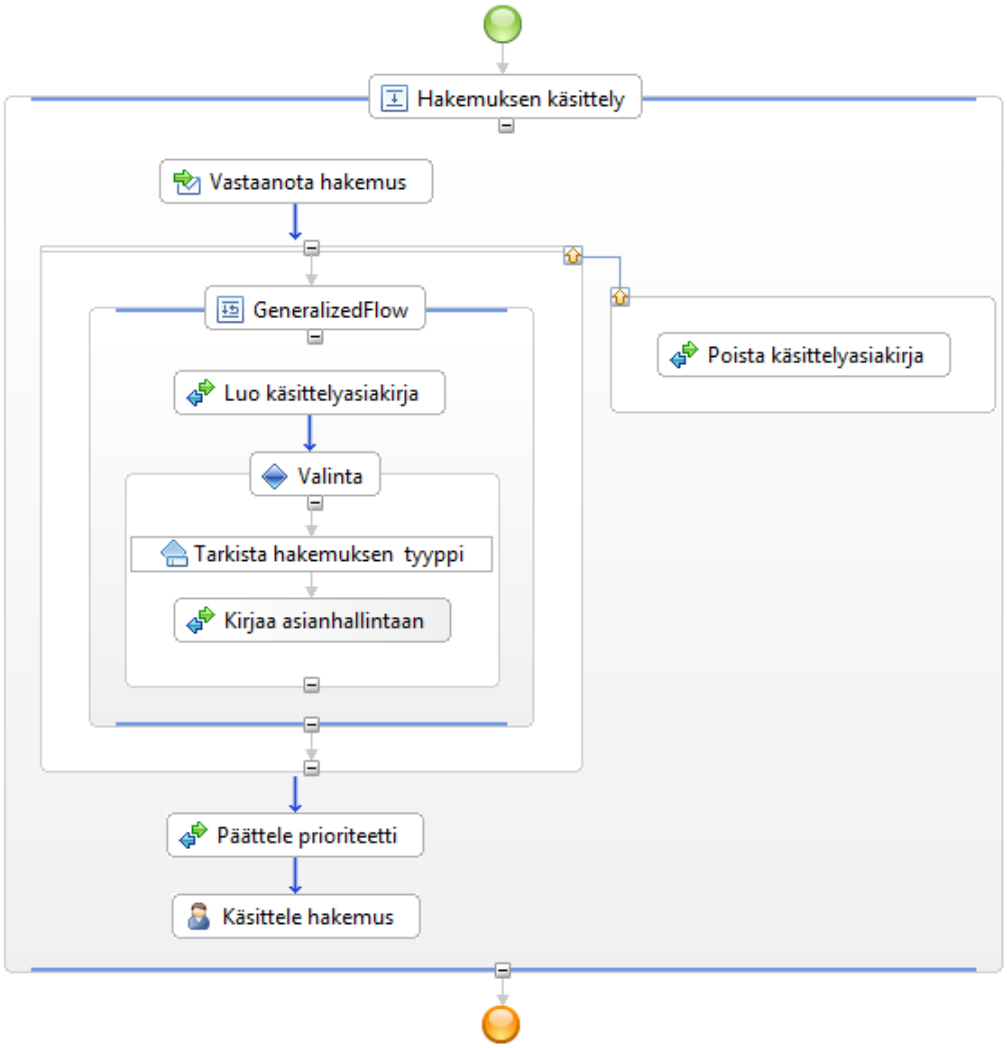
Palvelu	Toiminnot	Toteutustapa	Transaktio koordinoitavissa
Asiakirjapalvelu	luoAsiakirja, poistaAsiakirja	web-sovelluspalvelu	ei
Asianhallintapalvelu	kirjaa asia	tallennettu proseduuri	ei
Prioriteettipalvelu	päättele prioriteetti	SCA	kyllä

4.3.3 Koostepalvelun toteutus

Yksi koostepalvelun osatehtävistä on manuaalitehtävä, jonka suoritusaika voi vaihdella minuuteista päiviin tai viikkoihin. Tämän vuoksi koostepalvelu on toteutettava pitkäkestoisen transaktion asettamien vaatimukset huomioiden. Jokainen tehtävä muodostaa oman transaktionsa, joka vahvistetaan välittömästi sen suorittamisen päätyttyä. Palvelukoostaminen voidaan orkestroida BPEL-kielellä. BPEL-kielen rajoitusten vuoksi tiedon muunnoksissa eri tietomalleihin ja palvelukutsuissa käytettäville palveluille on kuitenkin käytännöllistä toteuttaa edustapalvelut ESB-ympäristön välityssovelluksena. Tällöin BPEL-orkestroinnissa voidaan keskittyä palveluiden kutsumiseen oikeassa järjestyksessä sekä kompensoinnin toteuttamiseen.

Kuvassa 15 on esitetty Integration Designer -ohjelmassa toteutettu koostepalvelun BPEL-orkestrointi. Tehtävien kutsuminen on toteutettu generalized flow -tyyppisen jäsentelyaktiviteetin sisään. Generalized flow ei ole osa BPEL-standardia, vaan se on tuotekohtainen laajennus, joka mahdollistaa muita jäsentelyaktiviteetteja enemmän vaihtoehtoisia rakenteita palvelujen kutsuun [21]. Koostepalvelun orkestroinnin vaiheet ovat seuraavat:

1. Vastaanota hakemus -vastaanottoaktiviteetti vastaanottaa koostepalveluun saapuvan sanoman ja tallentaa sen hakemusAsia-tyyppiseen BPEL-muuttujaan.



Kuva 15: BPEL-kielellä orkestroitu hakemuksen käsittely.

2. Tämän jälkeen suoritus siirtyy Luo käsittelyasiakirja -kutsuaktiviteettiin. Luo käsittelyasiakirja ja Kirjaa asianhallintaan -kutsuaktiviteetit ovat saman laajuusrajauksen sisällä. Laajuusrajaukseen liittyy kompensaatio, joka sisältää Poista asiakirja -kutsuaktiviteetin. Tämän avulla Luo asiakirja -kutsun suoritus voidaan kompensoida, jos Kirjaa asianhallintaan -kutsu epäonnistuu.

3. Kirjaa asianhallintaan -kutsu on sijoitettu Valinta-jäsentelyaktiviteetin sisälle, jolloin kutsu suoritetaan vain jos se täyttää asetetut ehdot, tässä tapauksessa ehdon hakemuksen tyypistä.

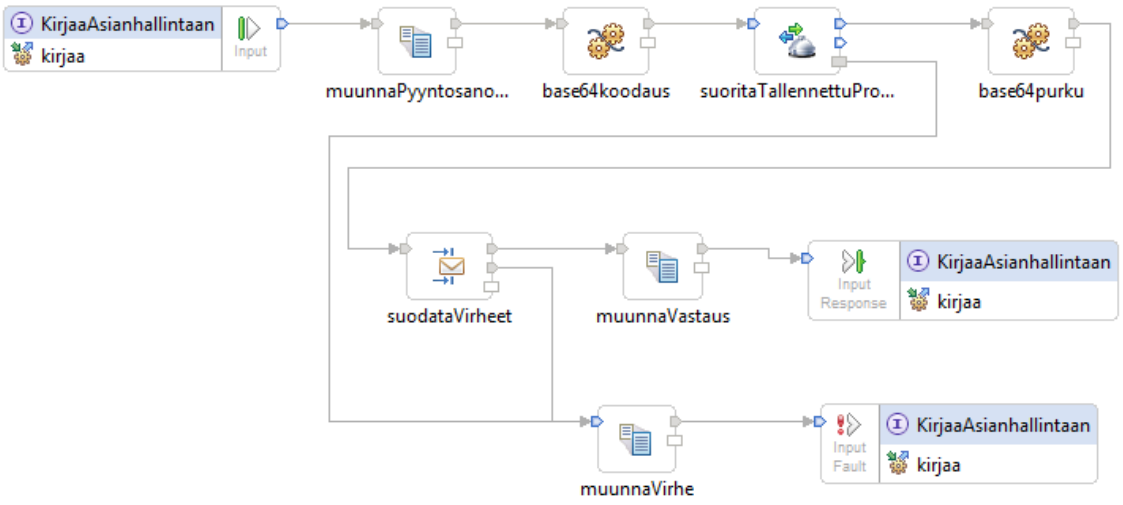
4. Lopuksi suoritetaan Päättele prioriteetti -palvelukutsu sekä Käsittele hakemus -manuaalitehtävä. Manuaalitehtävälle ei ole BPEL-standardissa määritelty toteutusta, joten palvelukoosteen kannalta ihmistoiminto vastaa epäsynkronista kutsuaktiviteettia.

Kaikki BPEL-orkestroinnin palvelukutsut tehdään paikallisiin edustapalveluihin.

Niiden kaikkien kutsu- ja vastaussanomat ovat hakemusAsia-tyyppiä. Edustapalvelut käyttävät ja tarvittaessa täydentävät hakemusAsia-tietotyypin tietueita, ja aktiviteetin suorituksen loputtua edustapalveluiden vastaussanomat tallennetaan takaisin BPEL-muuttujaan. Tästä seuraa, että BPEL-tasolla vältytään kaikilta tiedon muunnoksilta tietomallista toiseen ja koko BPEL-prosessissa tarvitaan vain yksi muuttuja. Edustapalvelut toteutetaan välitysmoduuleissa. Edustapalveluiden WSDL-rajapinnoissa määritellään palvelun toteuttava operaatio. Operaatiot ovat pyyntö-vastaus-tyyppisiä ja lisäksi ne sisältävät vähintään yhden virheterminaalin. Edustapalvelut sisältävät vähintään seuraavat vaiheet:

1. tiedon muuntaminen hakemusAsia-tietomallista palvelukohtaiseen tietomalliin
2. varsinaisen palvelutoteutuksen kutsuminen
3. virheentarkastelu, jos palvelutoteutuksella ei ole vahvasti tyypitettyjä virheterminaaleja
4. tiedon muuntaminen palvelukohtaisesta tietomallista hakemusAsia-tietomalliin tai virhesanoman tietomalliin

Edustapalveluista tarkastellaan lähemmin Kirjaa asianhallintaan -edustapalvelua. Varsinaisen palvelun toteutetaan tietokantasovelluksen tallennetulla proseduurilla. Tallennetun proseduurin kutsumiseksi on prosessiserveriympäristöön toteutettu SCA-komponenttiin JavaEE-sovellus, joka muodostaa yhteyden tietokantasovellukseen. Tässä esiteltävä edustapalvelu kutsuu tätä sovellusta, välittäen sille kutsusanomassa Base64-koodatun tallennetun proseduurin pyyntösanoman.



Kuva 16: Kirjaa asianhallintaan -edustapalvelun välistyssovellustoteutus.

Kuvassa 16 on esitetty Integration Designer -ohjelmalla toteutettu Kirjaa asianhallintaan -edustapalvelu. Edustapalvelun logiikka on toteutettu välitysvuo-tyyppiseen komponenttiin kehitysympäristön graafisella työkalulla. Edustapalvelu sisältää seuraavat vaiheet:

1. tiedon muunnos hakemusAsia-tietomallista tallennetun proseduurin pyyntösanomaksi
2. pyyntösanoman Base64-koodaus
3. kutsu tallennetun proseduurin suorittamiseksi
4. vastaussanoman Base64-purkaminen
5. vastaussanoman sisällön tarkastelu kutsun onnistumisen selvittämiseksi ja suorituspolun valinta tarkastelun perusteella
6. tiedon muunnos tallennetun proseduurin vastaussanomasta virhesanomaksi tai hakemusAsia-tietomalliin

Välitysmoduuli sisältää mahdollisuuden asettaa tilapäistä asiayhteystietoa (engl. transient context), jossa voidaan säilyttää dataa palvelukutsuista riippumattomasti koko välityksen suorituksen ajan. Tässä toteutuksessa asiayhteystietoon voidaan tallentaa tietomallin muunnoksen yhteydessä alkuperäinen hakemusAsia-objekti. Myöhemmässä muunnoksessa se taas palautetaan välityksen varsinaiseen hyötykuormaan, jota täydennetään palvelukutsusta saadulla tiedolla, tässä tapauksessa virityksen tulostiedolla.

4.4 Koostepalvelu 2: Laske maksut

Tässä aliluvussa kuvataan Laske maksut -koostepalvelulle asetetut vaatimukset ja esitellään koostepalvelun toteutus.

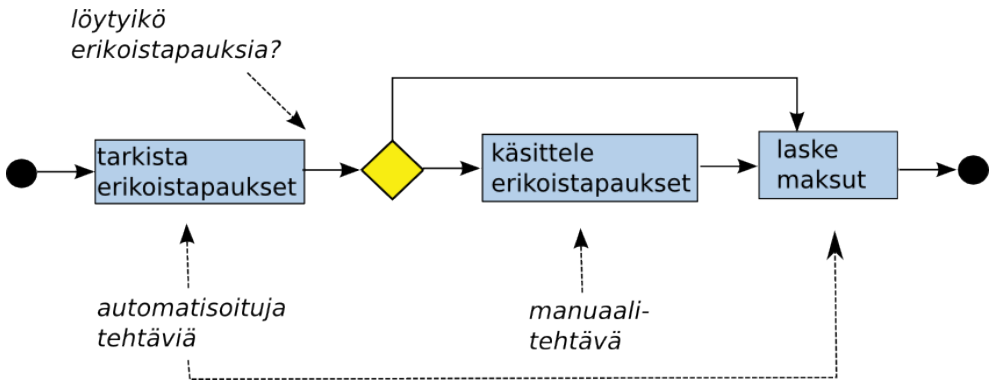
4.4.1 Palvelun kuvaus

Laske maksut -koostepalveluun sisältyy kolme erillistä tehtävää:

- tarkista maksujen erikoistapaukset
- käsittele erikoistapaukset
- laske maksut

Kuvassa 17 on esitettu koostepalvelun tehtävien suoritusjärjestys. Maksun erikoistapauksien tarkistaminen on koostepalvelun ensimmäinen tehtävä. Sen tulosten perusteella määräytyy, suoritetaanko toinen tehtävä, erikoistapausten käsittely. Viimeiseksi suoritetaan varsinainen maksujen laskeminen. Erikoistapausten tarkistaminen ja maksujen laskuun on käytettävissä oleva palvelu, erikoistapausten käsittely suoritetaan manuaalityönä.

Maksujen laskeminen kohdistuu tiettyyn hakemukseen, joka on vastaanotettu edellä kuvatulla koostepalvelulla. Nämä koostepalvelut ovat osa prosessinohjausjärjestelmää, jonka suoritusvuossa kulkee hakemusten yksilöintiin tarvittavat tiedot. Varsinainen käsittelyn kohteena oleva hakemus ei kulje prosessissa, vaan se haetaan tarvittaessa hakemusten hallintajärjestelmästä.



Kuva 17: Maksujen laskeminen.

4.4.2 Käytettävissä olevat palvelut

Tarkista maksun erikoistapaukset -tehtävän suorittamiseksi on tätä koostepalvelua varten kehitetty Java-ohjelmointikielellä SCA-komponenttiin toteutettu palvelu. Palvelukutsussa välitetään prosessin ohjaustiedot. Laske maksut -tehtävän suoritus tapahtuu kutsumalla IMS-transaktiosovellusta. Sovellus on käytettävissä vain tiettyinä ajanhetkinä. Palvelukutsussa välitetään sekä prosessin ohjaustietojen sisältämiä että hakemukselta saatavia tietoja. Varsinainen kutsu IMS-järjestelmään on SCA-komponenttiin toteutetussa Java-sovelluksessa, jonka tarjoamaa palvelua tässä koostepalvelussa käytetään hyväksi.

4.4.3 Koostepalvelun toteutus

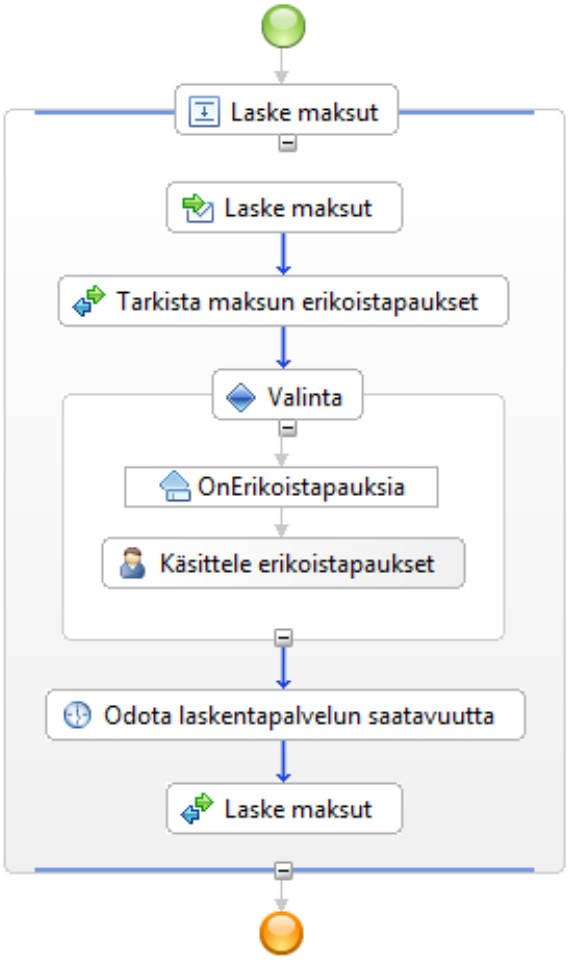
Tähänkin koostepalveluun sisältyy manuaalitehtävä, jonka suoritusajasta ei ole tietoa. Toisaalta palvelun osatehtävät eivät ole liiketoiminnalliselta luonteeltaan sellaisia, että ne pitäisi suorittaa samassa transaktiossa. Näiden seikkojen vuoksi koostepalvelu on helppo toteuttaa orkestroimalla palvelukutsut BPEL-kielellä. Koostepalvelun kuvauksesta sekä palvelujen ominaisuuksista selviää kaksi orkestroinnissa huomioon otettavaa seikkaa: erikoistapausten käsittelyn ehdollisuus sekä maksujen laskupalvelun oleminen käytettävissä vain tiettyyn vuorokaudenaikaan.

Kuvassa 18 on esitetty palvelun BPEL-orkestrointi. Edellä mainituista syistä palvelukutsut tehdään paikallisiin edustapalveluihin, joiden kaikkien pyyntö- ja vastaussanomat ovat samaa tyyppiä kuin koostepalvelun.

1. Laske maksut -vastaanottoaktiviteetti vastaanottaa koostepalveluun saapuvan sanoman ja tallentaa sen BPEL-muuttujaan.

2. Tarkista maksun erikoistapaukset -edustapalvelun kutsuaktiviteetti suoritetaan.

3. Siirrytään Valinta-jäsentelyaktiviteetin sisälle, jossa Käsittele erikoistapaukset -kutsuaktiviteetti suoritetaan, jos OnErikoistapauksia -tapausaktiviteetin ehto täyttyy.



Kuva 18: Laske maksut -koostepalvelun BPEL-orkestrointi.

4. Odotusaktiviteettiin on määritelty kalenteri, johon määriteltyjen aikojen puitteissa orkestroinnin suorituksessa edetään.

5. Laske maksut -kutsuaktiviteetin suorittaminen.

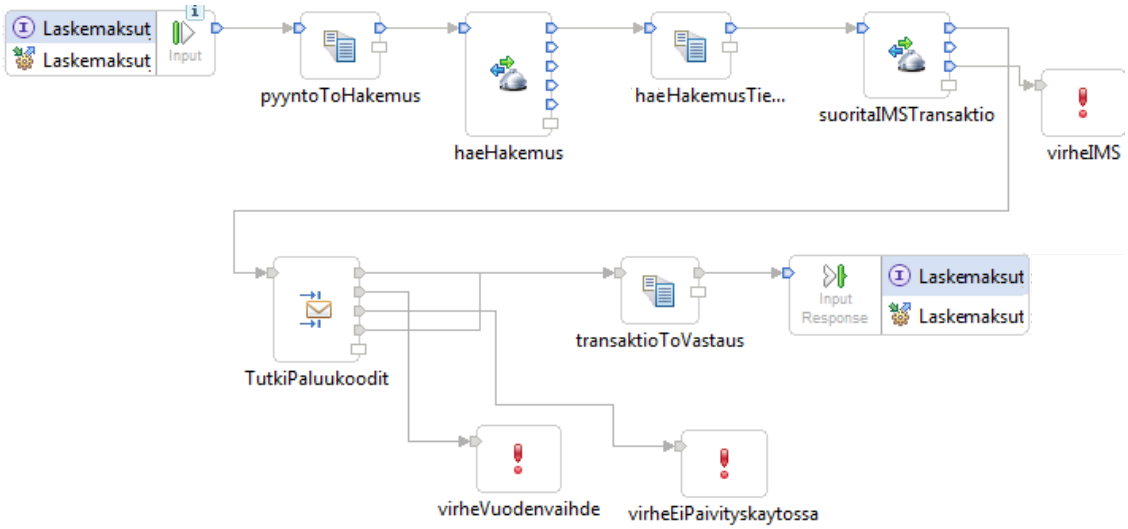
Laske maksut -tehtävän suorittamiseksi kutsuttavaan transaktiosovelluskutsun tarvitaan myös hakemuksen sisältämiä tietoja. Ennen transaktiosovelluskutsua on siis tehtävä kutsu hakemushallintajärjestelmään, joten Laske maksu -edustapalvelusta muodostuu oma koostepalvelunsa.

Kuvassa 19 on Laske maksut -tehtävän edustapalvelun toteutus välitysmoduulina. Se sisältää seuraavat vaiheet:

1. tiedon muunnos prosessin ohjaustiedoista hakemushakukutsun tietomalliin

2. kutsu hakemuksen hakuun

3. tiedon muunnos transaktiosovelluksen kutsussa käytettävään tietomalliin



Kuva 19: Laske maksut -edustapalvelun välityssovellustoteutus.

4. kutsu transaktiosovellukseen

5. vastaussanoman sisällön tarkastelu operaation tuloksen selvittämiseksi ja välityksen suorituspolun valinta sen perusteella

6. tiedon muunnos edustapalvelun vastaussanomaksi tai suorituksen päättyminen virheeseen

Alkuperäinen pyyntösanoma talletetaan ensimmäisen muunnoksen yhteydessä välityssovelluksen tilapäiseen asiayhteystietoon. Välitysvuon suorituksen päätyminen virhe-komponenttiin, kuten kuvassa 19 näkyvä virheIMS, johtaa siihen että välityskomponentti palauttaa virheen BPEL-orkestraatioon.

4.5 Toteutusratkaisujen arviointi

Koostepalvelut on toteutettu orkestroimalla tehtäviä toteuttavien palvelujen kutsut. Koostepalvelujen prosessikuvauksissa esitettyjä tehtävien suoritusjärjestystä vastaava rakenne on onnistuneesti saatu aikaan käyttämällä BPEL-kielen toimintoja. Kehitysympäristön graafinen BPEL-notaatio mahdollistaa prosessikuvausten tehokkaan toteuttamisen. Graafinen esitys mahdollistaa myös rakenteen helpon tarkastelun. Toteutusympäristön ominaisuuksien seurauksena koostepalveluille generoituu myös julkaistavissa oleva WSDL-rajapinta. Koostepalvelujen käyttämiä palveluja varten toteutetut edustapalvelut noudattavat palvelusuuntautuneen arkkitehtuurin periaatteita.

Tutkitaan Koostepalvelu 1:n transaktionaalisuutta tarkastelemalla transaktioille määriteltyjä ominaisuuksia: 1. jakamattomuus, 2. ristiriidattomuus, 3. erillisyys ja 4. pysyvyys.

1. Koostepalvelun automatisoiduista tehtävistä kaksi on luonteeltaan tietojärjestelmän tilaa muuttavia: asiakirjan luonti ja asianhallintaan kirjaus. Asiakirjan

luontiin on olemassa sen tuloksen kompensoiva toiminto, poista asiakirja. Sijoittamalla nämä ratkaisussa kuvatulla tavalla BPEL-laajuusrajauksen sisälle ja määrittelemällä sille kompensaatio, toteutuu näiden kahden tehtävän muodostamalle kokonaisuudelle jakamattomuus. Näitä seuraavat luku-tyyppinen automatisoitu tehtävä ja manuaalitehtävä eivät ole jakamattomuuden piirissä.

2. Koostepalvelun ristiriidattomuus toteutuu, sillä kirjoittavissa toiminnoissa hyödynnetään palvelujen tarjoamia operaatioita. Niiden oletetaan toimivan omien eheysrajoitteidensa puitteissa.

3. Tämä toteutusratkaisu ei toteuta erillisyyttä, sillä jokaisen tehtävän tulokset ovat näkyvissä ulkopuolisille tahoille välittömästi niiden suorituksen jälkeen. Erillisyyden toteuttaminen tämänkaltaisissa pitkäkestoisissa liiketoimintapalveluissa ei ole mahdollista käytettävissä olevilla toteutustekniikoilla.

4. Koostepalvelun pysyvyys toteutuu pitkälti suoritusympäristön toiminnallisuuden puitteissa. Prosessiserveri tallentaa prosessin tilan pysyvään muistiin, joten tilatietoa ei menetetä järjestelmän kaatuessa. Palvelujen suoritusjärjestelmät osaltaan toteuttavat tehtävien tulosten pysyvyyden.

Taulukossa 3 on vedetty yhteen Koostepalvelu 1:n toteutuksen transaktionaaliset ominaisuudet.

Taulukko 3: Koostepalvelu 1:n transaktio-ominaisuudet

Transaktio-ominaisuus	Toteutuu koostepalvelussa
jakamattomuus	osittain
ristiriidattomuus	kyllä
eristyvyys	ei
pysyvyys	kyllä

Koostepalvelu 2 ei sisällä tehtäviä, joille olisi määritelty kompensoivia operaatioita. Jokainen tehtävä on siis puhtaasti omassa transaktiossaan. Koostepalvelun vaativin osa on saada Laske maksut -tehtävän legacy-toteutus osaksi koostepalvelun transaktionhallintaa. Edustapalveluun toteutettu sanoman sisällön tarkasteluun perustuva virheentarkastelu toteuttaa tämän palauttamalla virhesanoman, jos transaktion suorittaminen ei onnistunut. Sanoman sisältöön perustuva virhe on tällöin onnistuneesti tyypitetty orkestrointitoteutuksen ymmärtämäksi virhetyypiksi.

4.6 Yhteenveto

Tässä luvussa on kuvattu kaksi koostepalvelutoteutusta. Koostepalvelun koordinaatio tapahtuu orkestrointi-paradigman mukaisesti ja on toteutettu käyttäen BPEL-kieltä. Koostepalvelujen ominaisuuksien vuoksi niiden transaktionaalisuus on toteutettu pääosin hyödyntämällä kompensaatiota. Tuloksena syntynut prosessikuvaus

on suoritettavissa siihen soveltuvalla prosessimoottorilla. BPEL tukee heikosti monimutkaista tietomuunnosta, jonka vuoksi koostepalvelutoteutuksiin on toteutettu paikalliset edustapalvelut. Edustapalvelut on toteutettu palveluväylätoteutuksen välityssovelluksilla, joissa on monipuoliset tietomuunnokseen tarvittavat toiminnot. Tällä toteutustapojen yhdistelmällä pystytään toteuttamaan koostepalvelu, joka toimii liiketoiminnallisesti oikealla tavalla sekä takaa palvelun transaktionaalisuuden riittävällä tasolla.

5 Johtopäätökset ja yhteenveto

Hajautunut tietojärjestelmäympäristö muodostuu toteutustekniikaltaan erilaisista ja mahdollisesti eri organisaatioiden hallitsemista tietojärjestelmistä. Palvelusuuntautunut arkkitehtuuri on integraatioarkkitehtuuri, joka soveltuu tälläisen hajautuneen ympäristön integroimiseen silloin, kun ympäristön järjestelmien toiminnallisuuksia halutaan käyttää osana prosessinohjausjärjestelmää. Liiketoimintaprosessien automatisoinnissa keskeisessä osassa ovat koostepalvelut. Koostepalveluilla muodostetaan kokonaisuuksia, jotka automatisoivat liiketoimintaprosesseja, tai niiden osia. Koostaminen tapahtuu tekemällä koordinoidusti kutsuja tietojärjestelmien palveluihin. Palvelut toteuttavat tiettyjä liiketoiminnallisia tehtäviä.

Hajautuneeseen tietojärjestelmään sopivin koostepalvelujen koordinaatiotapa on palvelukutsujen orkestrointi. Orkestraatiossa koosteen logiikka on keskitetty koordinaattoriin. Koordinaattorin toteuttamista varten koostepalvelun liiketoimintalogiikka on pystyttävä kuvaamaan riittävällä ilmaisuvoimalla ja tarkkuudella. Vaikka orkestrointi voidaan toteuttaa millä tahansa ohjelmointikielellä, tehokkainta on käyttää erityisesti siihen kehitettyä kieltä. Yleinen ohjelmointikieli on BPEL, joka voidaan suorittaa siihen soveltuvalla prosessimoottorilla.

Koostepalvelun transaktion toteuttaminen vaatii sen osatehtävien transaktioiden käsittelyä hajautettuna transaktiona. Hajautetun transaktion hallinta vaatii yhteisen transaktiokoordinaattorin käyttöä. Transaktiokoordinoinnissa käytetyin koordinaatiotapa kaksivaiheinen vahvistamiskäytäntö. Transaktiokoordinaattorin käyttö vaatii kuitenkin palveluilta teknisiä edellytyksiä osallistua sen toimintaan. Heterogeenisessä tietojärjestelmäympäristössä transaktiokoordinaattorin käyttö voi olla teknisesti mahdotonta, tai käytön saattaa estää hallinnolliset syyt, sillä koordinaattorin käyttö edellyttää siihen osallistuvien transaktioiden hallintaoikeuksien luovuttamista koordinaattorille.

Koostepalvelun luonne, kuten sen pitkäkestoisuus, saattaa estää transaktiokoordinoinnin suorituskykyongelmien vuoksi. Tämä johtuu siitä, että ACID-määreiden noudattaminen vaatii alitransaktioiden resurssien lukitsemista. Resurssien lukitseminen pitkäkestoisen koostepalvelun keston ajan on järjestelmien suorituskyvyn kannalta mahdotona. Transaktiokoordinaattorin käyttämisen sijaan, pitkäkestoisen koostepalvelun transaktion toteuttaminen on otettava huomioon koostaamisen suorituslogiikan suunnittelun yhteydessä. Osana orkestrointikuvausta on mahdollista määritellä kompensoivia tehtäviä, jotka suoritetaan jos koostepalvelun suoritus keskeytyy. Tällöin jo suoritettujen tehtävien tulokset voidaan loogisesti kumota suorittamalla kompensoivat tehtävät. Erityisesti on pyrittävä tilanteeseen, jossa koostepalvelun kaikkiin tehtäviin liittyy kompensoiva tehtävä, jolloin koostepalvelun transaktion jakamattomuus toteutuu. Tämän toteuttamiseksi palveluissa on oltava saatavilla kompensoivia toimintoja. Tämä on erityisen tärkeää silloin, kun transaktiokoordinaattorin tukeminen ei teknisistä tai liiketoiminnallisista syistä onnistu.

Koostepalvelun suorituslogiikan kuvaaminen kompensaatioineen onnistuu BPEL-kielen avulla. Se ei kuitenkaan mahdollista vaativien datatransformaatioiden toteuttamista, joka saattaa olla tarpeellista muodostettaessa palvelujen pyyntösanomia ja tulkittaessa vastaussanomia. Tästä syystä BPEL-prosessikuvaus vaatii monesti pai-

kallisen edustapalvelun, joka huolehtii monimutkaisista datatransformaatioista. Näitä edustapalveluja voidaan toteuttaa esimerkiksi palveluväyläympäristöön välityssovelluksina.

Käytännössä, monissa tapauksissa koostepalvelun pitkäkestoisuus tai käytettävissä olevien palvelujen rajoitteet estävät koordinaatioprotokollan käytön. Tyypillisesti koostepalvelua toteuttavalla taholla on rajalliset mahdollisuudet vaikuttaa palvelujen ominaisuuksiin. Tämän vuoksi uusia palveluja toteutettaessa on harkittava myös palvelun kyky osallistua hajautettuihin transaktioihin, vaikka sille ei olisi välitöntä tarvetta. Käytännössä tämä tarkoittaa sellaisen toteutustekniikan käyttöä, joka pystyy osallistumaan hajautettuun transaktionhallintaan. Palvelusuuntautuneessa arkkitehtuurin mukaisessa ympäristössä kirjoitushetkellä parhaiten tuettu ja siten hyödyllisin toteutustekniikka on web-sovelluspalvelu ja siihen liittyvä WS-Transaction -koordinaatioprotokolla.

Täysin kompensoitavissa olevaa transaktioketjua kutsutaan saagaksi. Saaga täyttää transaktion jakamattomuuden periaatteen, mutta sen suoritus ei ole eristetty muista transaktioista. Pitkäkestoisten transaktion eristyvyyden toteuttamiseksi ei ole vakiintunut keinoa, eikä tiedossa ole laaja-alaista, standardointiin tähtäävää ratkaisuyritystä. Eristyvyyden takaaminen esimerkiksi web-sovelluspalvelujen koordinoidussa ja pitkäkestoisessa hajautetussa transaktiossa lisäisi niiden soveltuvuutta liiketoiminnallisesti kriittisiin ohjelmistoihin merkittävästi.

Viitteet

[1] Laamanen, K. *Prosessijohtamisen käsitteet.* Teknologiateollisuus Oy, 2009.

[2] Smith, A. *An Inquiry into the Nature and Causes of The Wealth of Nations.* Verkkodokumentti. [Viitattu 23.01.2013] Saatavissa: http://www.marxists.org/reference/archive/smith-adam/works/wealth-of-nations/index.htm

[3] Juric, M. et al. *WS-BPEL 2.0 for SOA composite applications with IBM WebSphere 7 : define, model, implement, and monitor real-world BPEL 2.0 business processes with SOA-powered BPM.* Birmingham, UK, Packt Publishing Ltd., 2010.

[4] *Web Services Business Process Execution Language Version 2.0 - OASIS Standard.* OASIS. Verkkodokumentti. [Viitattu 26.07.2012] Saatavissa: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

[5] *Business Process Execution Language.* Wikipedia. Verkkodokumentti. [Viitattu 26.07.2012] Saatavissa: http://en.wikipedia.org/wiki/Business_Process_Execution_Language.

[6] *XML Path Language (XPath) Version 1.0 W3C Recommendation 16.* W3 Consortium. Verkkosivusto. [Viitattu 02.08.2012] Saatavissa: http://www.w3.org/TR/xpath/

[7] Keen, M. et al. *Patterns: Implementing an SOA Using an Enterprise Service Bus.* IBM International Technical Support Organization, 2004.

[8] Endrei, M. et al. *Patterns: Service-Oriented Architecture and Web Services.* IBM International Technical Support Organization, 2004.

[9] Rosen, M et al. *Applied SOA : Service-Oriented Architecture and Design Strategies.* Wiley, 2008.

[10] Rayns. C et al. *CICS and SOA: Architecture and Integration Choices.* IBM International Technical Support Organization, 2012.

[11] Jäntti. J et al. *IMS Connectivity in an On Demand Environment: A Practical Guide to IBM Connectivity.* IBM International Technical Support Organization, 2006.

[12] Erl, T. *SOA Design Patterns.* Prentice Hall, 2008.

[13] Kifer. M. et al. *Database Systems: An Application-Oriented Approach.* 2. painos. Pearson Education, Boston, 2006.

[14] Garcia-Molina, H., Salem, K. *Sagas* Vol. 16, no. 3. ACM, 1987.

[15] Gray, J. *The transaction concept: Virtues and limitations.* Proceedings of the Very Large Database Conference, 1981.

[16] Ezenwoye, O., Sadjadi, S. *Composing Aggregate Web Services in BPEL.* ACM, Proceedings of the 44th annual Southeast regional conference, s. 458-463. New York, 2006.

[17] Vinoski, S. *Integration with Web Services.* IEEE Internet Computing, Vol. 7, Iss. 6, s.75-77, 2003.

[18] Booth, D. et al. *Web Services Architecture.* W3C Working Group Note 11, 2004.

[19] Juric, M. et al. *SOA Approach to Integration.* Packt Publishing Ltd., 2007.

[20] Hofstede, A. et al. *Modern Business Process Automation. YAWL and its Support Environment.* Springer Berlin Heidelberg, 2010.

[21] *Working with generalized flow activities.* Verkkosivusto. [Viitattu 8.1.2013] Saatavissa: http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/index.jsp?topic=/com.ibm.wbit.620.help.bpel.ui.doc/topics/tcyclic.html

[22] *WS-I Deliverables.* Verkkosivusto. [Viitattu 8.1.2013] Saatavissa: http://ws-i.org/deliverables/Default.aspx

[23] Putte, G. ja Gavin, L. *Technical Overview of WebSphere Process Server and WebSphere Integration Developer.* International Technical Support Organization, 2005.

[24] *Mediation modules.* IBM. Verkkosivusto. [Viitattu 10.1.2013] Saatavissa: http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/topic/com.ibm.websphere.wesb.doc/concepts/cwesb_mediationmodules.html

[25] Edwards, M. *Service Component Architecture (SCA).* OASIS. Verkkosivusto. [Viitattu 10.1.2013] Saatavissa: http://oasis-opencsa.org/sca

[26] den Haan, J. *What every architect should now know about the Service Component Architecture (SCA).* Verkkosivusto. [Viitattu 11.1.2013] Saatavissa: http://www.theenterprisearchitect.eu/archive/2009/03/11/

[27] Choi, S. et al. *Maintaining Consistency Under Isolation Relaxation of Web Services Transactions.* Lecture Notes in Computer Science, Vol. 3806, s.245-257, 2005.

[28] Josefsson, S. *The Base16, Base32, and Base64 Data Encodings.* RFC 4648, IETF, 2006. Saatavissa: https://tools.ietf.org/html/rfc4648

[29] Silberschatz, A. *Database System Concepts.* 5. painos. New York, USA. McGraw-Hill, 2006.

[30] *What is Middleware.* OW2 Consortium. Verkkosivusto. [Viitattu 8.2.2013] Saatavissa: http://middleware.objectweb.org/

[31] Chang, J. *Business Process Management Systems: Strategy and Implementation* Boca Raton, USA. Auerbach Publications, 2006.

[32] *Data flow modeling for application development.* IBM. Verkkodokumentti. [Viitattu 26.07.2012] Saatavissa: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.btools.help.modeler.doc/doc/concepts/widmodeling/dataflowmodeling.html

[33] *Simple Object Access Protocol (SOAP) 1.1* W3C. Verkkodokumentti. [Viitattu 01.03.2013] Saatavissa: http://www.w3.org/TR/soap/

[34] *YAWL: Yet Another Workflow Language.* YAWLGroup. Verkkosivusto. [Viitattu 01.03.2013] Saatavissa: http://www.yawlfoundation.org/

[35] *Oracle BPEL Process Manager.* Oracle. Verkkosivusto. [Viitattu 7.03.2013] Saatavissa: http://www.oracle.com/technetwork/middleware/bpel/overview/index.html

[36] *WebSphere Enterprise Service Bus.* IBM. Verkkosivusto. [Viitattu 7.03.2013] Saatavissa: http://www-01.ibm.com/software/integration/wsesb/

[37] *Microsoft BizTalk Server.* Microsoft. Verkkosivusto. [Viitattu 7.03.2013] Saatavissa: http://www.microsoft.com/biztalk/en/us/default.aspx

[38] *Oracle Service Bus Overview.* Oracle. Verkkosivusto. [Viitattu 11.03.2013] Saatavissa: http://www.oracle.com/technetwork/middleware/service-bus/overview/index.html

[39] *Apache Synapse Enterprise Service Bus (ESB).* Apache Server Foundation. Verkkosivusto. [Viitattu 11.03.2013] Saatavissa: http://synapse.apache.org/

[40] *JBoss ESB.* JBoss Community. Verkkosivusto. [Viitattu 11.03.2013] Saatavissa: https://www.jboss.org/jbossesb/

[41] *OASIS Web Services Transaction (WS-TX) TC.* OASIS. Verkkosivusto. [Viitattu 13.03.2013] Saatavissa: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-tx

[42] Tripathi, M. *SCA (Service Component Architecture) - Part 1.*Verkkodokumentti. [Viitattu 13.03.2013] Saatavissa: http://softwarearchitectureandarchitects.blogspot.fi/2011/09/lets-briefly-talk-about-sca-part-1.html

[43] *IMS Family.* IBM. Verkkosivusto. [Viitattu 13.03.2013] Saatavissa: http://www-01.ibm.com/software/data/ims/

[44] *Java(TM) Transaction API (JTA) Specification 1.0.1.* Oracle. Verkkodokumentti. [Viitattu 13.03.2013] Saatavissa: http://download.oracle.com/otndocs/jcp/7286-jta-1.0.1-spec-oth-JSpec/

[45] *CICS Family.* IBM. Verkkosivusto. [Viitattu 13.03.2013] Saatavissa: http://www-01.ibm.com/software/htp/cics/

[46] *Distributed TP: The XA Specification.* Verkkodokumentti. [Viitattu 13.03.2013] Saatavissa: https://www2.opengroup.org/ogsys/catalog/c193